START COMPUTING on the AMSTRAD

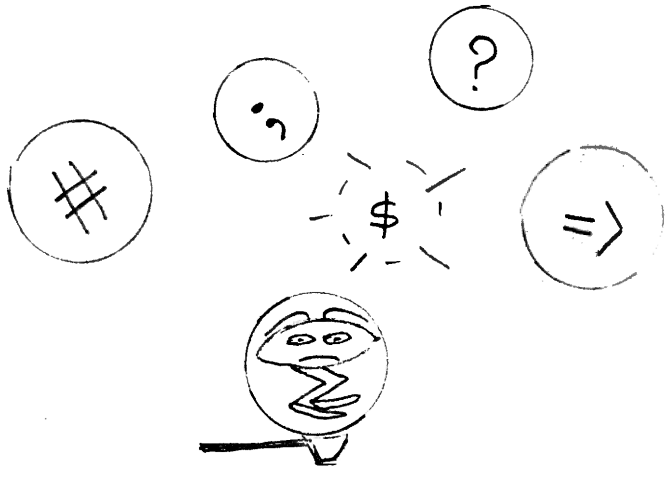
CPC6128

JUDITH THAMM

a basic course for beginners

CONTENTS

INDEX TO PROGRAM



INTRODUCTION

No matter what age you are or what your level of education, computing is as easy as learning to ride a bike. The few 'Road Rules' you need to know- and you don't need to know many, are easy to pick up.

The greatest pain in the neck is that other people will persist in talking in Computer Jargon, leaving you out in the the cold. Essential 'Jargon' and explanations are dealt with later so that some will rub off on you.

I hope my explanations anticipate what you want to know as you progress. In case you find explanations too obvious, remember this is written for those who have never typed before, those who thought this was just a games machine, and those who do a little and come back much later to try some more...

If you have never typed, a Typing Tutor is a great way to learn where the keys are. But unlike an ordinary table, a computer table must never share space with a cuppa- one splash on the keyboard can be very expensive and leave you without a computer for some time!

I must confess that I am a late starter. I suffer from a computer-wise teenage son, who declared I would never learn how to use the computer....

We had barely joined an Amstrad Club when the honour(?) of being secretary was thrust upon me. I found I was helping beginners and writing out instruction sheets just to save repeating myself. Then I'd think of a better way to explain something, and suddenly I found I was writing an instruction book.. My first efforts are still being used by those who are getting started. This is a set of new programs designed to give you a hankering for more.

My approach is different from anything I could find to help me when I was an absolute beginner. I couldn't find a book that would answer the questions I wanted answered. My only help was from a patient shop owner and equally patient club members. Now I consider myself an advanced beginner !

I have tried to put in very detailed explanations. I hope they help you with the answers to those questions you hesitate to ask in case you make a fool of yourself.

If you bought your Amstrad for hobby or for business, don't be put off by the User Instruction Book. It was written by the people who know all about making computers. Real experts in computing often take the simplest things for granted, and this is the case with the Manual- the explanations need explanations if you are a beginner!

I'm hopeless at Zap The Whatsit games. I'm too busy watching the animation on screen. Perhaps if I'd grown up with a cartoon creating machine, I'd be less intrigued with the animation and more able to concentrate on the game.

So jump in, fingers flying, and in a month or two, you'll wonder why you ever thought you'd never be able to use a computer!

The first thing someone will want to know is 'What can you do with the computer?' Have a formatted disc ready (see Chapter One). Then, type in this short program. No explanations yet, but this is one of those things you'll want to be able to do right now. You won't be ready to understand the why's and wherefore's of this just yet. Watch out for capitals, $, <>, brackets and spaces.

PRESS ENTER AFTER EACH LINE.
CHECK EACH LINE TWICE BEFORE YOU PRESS ENTER !!!
It's much harder to correct errors than get it right first time. Line 50 will take more than one row of screen. Your screen and the program here should look identical. Do not press RETURN until all of line 50 is typed. There is 1 space after 'going '. Explanations later.

```
10 REM - a scroll
20 MODE 1
30 INK 0,18
40 INK 1,8
50 a$="I'm starving !... When are we goi
ng to eat ?... "
60 b$=a$
70 WHILE INKEY$<>" "
80 IF LEN(b$)<40 THEN b$=b$+a$
90 LOCATE 1,1
100 PRINT LEFT$(b$,40)
110 b$=RIGHT$(b$,LEN(b$)-1)
120 WEND
130 END
```

(If you press a key the program will stop.)

SAVE "SCROLL then try it out Type: RUN

To use the program again from the disc, type: RUN "SCROLL
The program is explained in Chapter 14.

Chapter 16 is about making music on the computer and can be attempted at any stage you like.

I hope you enjoy Learning Basic on the Amstrad CPC6128.

CHAPTER ONE HANG ON AND START PEDALLING...

Start by connecting the three cables between the monitor and keyboard. (Chapter 1,page 2 of the Manual). Pick up the keyboard and tilt it forward or turn it around to read all the names of the pieces on the back. Connect the cables THEN plug in the cord and turn on the power.

Get into some good habits. SWITCH ON the Monitor first and then the Keyboard. SWITCH OFF the Keyboard first and then the Monitor. Do not leave one or the other on. Think of the computer as another television set with a message keyboard attached.

The Monitor switch is the one that says 'POWER', the brightness control is the knob you can twiddle on the right side of the Monitor.

As you sit at the keyboard, the ON/OFF switch is to the right of the cables. Left is ON, right is OFF. The volume control (below the red 'ON' light) is LOUDER to the left and softer to the right.

If the screen does not light up when you turn the computer on, check that the power is on, check that the keyboard switch is to the left and then check that the power switch on the Monitor is pushed in (IN for ON, OUT for OFF). Of course you did check that the three cables are all pressed in firmly ...

You should have bought some blank discs. You really DO need 3. One to SAVE your programs onto, and two to make working copies of your CP/M Master discs. That wasn't just hard sell on the part of the shop assistant if you were persuaded to buy several blank discs. If you weren't sold at least 3 extra blank discs, you will need to buy them now.

If you are unlucky enough to have a school age (or worse a teenage) member in your family, you will have been persuaded to buy a game too. (Did I say 'a' game? Sorry about that slip. I should have spelt the word as 'games' with a capital 'S'.)

Here comes the hard part for an adult to have to swallow. By now, child prodigy will have put on a game and be galloping through it... See how easy it is? Now it's the adult's turn. First move in the game and he's dead! Unless it's a game such as Gauntlet, stop! or you'll be sorry..! If you don't want an inferiority complex, just watch the game to begin with. You can get an idea of what to do and then save your playing in front of a critic until you've done some sneaky practising.

Another good habit is to HAVE THE DISC DRIVE EMPTY WHEN YOU SWITCH ON OR OFF. You don't have much trouble with the first disc drive, but it is when you get a second disc drive that you can lose the contents of a disc if you switch on or off with the disc in the drive.

Try using the computer now.

Turn on the computer and wait until you see the Amstrad title and the copyright lines, Basic 1.1 and the READY sign with the square CURSOR underneath. The red lights will be on at the top of the keyboard and by the disc drive. (No disc needed yet.)

The READY sign is the 'prompt' to tell you it is your turn to do something and the CURSOR (the solid yellow square) shows you where the writing is going to go on the screen.

Every key you press makes a CHARACTER- Jargon for letter, number, space or symbol. The cursor moves one position ahead as you type in each character. A space is recorded as a 'space' character when you leave a space between words, just as a letter is recorded as a 'letter' character.

You will need to know the keyboard. It is different from that of a typewriter for instance, as each key does several jobs, besides producing letters in upper (capitals) or lower case (small letters). They can print a set of shapes or be made to produce new shapes. As well, certain combinations of keys do special things. Pressing CONTROL, SHIFT, ESC in that order to reset or clear out the computer's memory.

A QUICK TOUR OF THE KEYBOARD

Read this through and then refer back to it as you need to. Don't try to take it in all at once. Some things mentioned you may never use.

CONTROL Used in word processor and other advanced programs to give a key another function beside the one that is shown on it eg CONTROL and P in CP/M will send the text on the screen to the printer.

CONTROL SHIFT ESC (Press together.) Clears the screen and the memory of the computer and brings back the start up screen and messages, ready for the computer to work in BASIC, the easiest computer language. You will use this often.

SHIFT Held down gives capital letters (UPPER CASE) and the upper row of symbols on other keys. All the lower symbols and small letters (LOWER CASE) are ready for use when the computer is switched on.

CAPS LOCK Press to turn on. [Like on a typewriter] Turns the alphabetical keys into UPPER CASE - capital letters. Press again to return to LOWER CASE. Does not change any other keys eg punctuation.

CONTROL CAPS LOCK. Together, they give UPPER CASE to all but the numbers on the keypad and the arrow keys. Press all three at

once to turn on, and press all three together again to turn off.

TAB The TABulator lets you set out work from a point other than the far left side of the screen, (like on a typewriter). Unless it is used in a word processor program, you must include the instructions for its use in a program.

ESC Pressed once, it pauses a program when you are LISTing it. Pressed twice it allows you to interrupt a LISTing or stop it in order to make an alteration.

! [above 1] Exclamation mark. Used in some Basic commands eg LOAD "! dots.bas where you are loading from tape.

" [above 2] Double speech marks (quotation marks) are used to show that something is to be printed on the screen: eg

PRINT "My computer is easy to use."

Whatever is between the speech marks will appear that way on the screen when RETURN is pressed.

[above 3] Called a HASH sign and is used to tell the computer where information is to be sent. It may be to a piece of the screen set aside for messages ie a WINDOW. The DEFAULT WINDOW is the whole screen when you first switch on- its name is #0. You can have 8 WINDOWs, ie from #0 to #7 on the screen at once. LIST #8 will send a program listing to the printer and print it. PRINT with a #8 beside it is a printer command, and anything with #9 is to be stored on or retrieved from disc or tape.

$ [above 4] Beside being a dollar sign this is a STRING sign. A string can be a group of numbers or letters to be referred to in a code: eg

t$=total : PRINT t$ [Press RETURN] [You can not say total=t$.]

And the computer will print the word 'total'.

% [above 5] Besides a percent sign, % is used to store numeric information in a space saving code. The number is 'rounded off' to the nearest whole number eg t%

& [above 6] Used to indicate that a hexadecimal number is to follow eg the hexadecimal number for & is &H26. [hexadecimal is shortened to HEX.] If machine code is added to a basic program, HEX is used.

' [above 7] The apostrophe must be used instead of quotation marks in PRINT statements. Used instead of REM- a comment line.

= [above hyphen] Means not only equals, but is often used to show that something has now been given a new name: eg

IF a=10 THEN a=b

means that when 'a' gets to number 10, it is to be called 'b'.

↑ [below £] Used to find the power of another number.
eg 3↑3 = 27, (3*3*3), 3↑4 = 81 (3*3*3*3) [Expotentation]

CLR Removes the character under the cursor and if held down, swallows characters to the right of the cursor.

DEL Removes the character to the left of the cursor and if held down, wipes out all characters to the left.

¦ [Sign above the @, next to p.] Called a BAR. eg ¦CPM boots (jargon for loading up) CP/M which then allows access to the programs on the discs that came with the computer.

O Note that zero has a slash through it on the keyboard, on screen, and in LIST #8 printouts, but in this (written using a word processor program), zero is oval (zero 'O') and capital letter 'O' is square.

* Asterisk and the Multiply sign. Also used as a wild card sign meaning 'all'.
¦ERA,"*.BAK means erase 'all' back up programs or files with a name ending in .BAK . '*' is then called a wildcard.

/ [under ?] Divides things normally.
\ divides and rounds off to a whole number. [The backslash.]

? Question mark and short for 'PRINT'. eg.

? "hello" is the same as PRINT "hello".

f keys Use either the keys at the top of the keyboard or the f keys for calculations. The f keys (function keys) can be used in a program (called defining a key) to do special functions eg f1 could be made to print a special word each time it was pressed. Then that key could not be used as a 'number' key until it was changed back to normal or the computer reset.

ARROW KEYS Used to Move the cursor around the screen in the direction they point.

: The colon is used to mark the end of a computer sentence. This saves using a new number line in a program for another instruction. eg INK 0,1:INK 1,6:INK 2,3 [3 'sentences' in 1]

; The semi-colon tells the computer to do something next to another group of eg numbers, on the same line, instead of going to a new line. It also stops control codes from being printed on screen [see explanations later when using TAG].

< Means less than, and points to the Left (remember: Less points to the Left).

> Means more than, and points to the right (remember: More points to the Right).

<> Means not equal to.

n Probably the most used variable. 'n' is often used to indicate a number. If a program says eg FOR n=1 TO 3, it means 'n' (code for number) will change in value (vary, hence the term variable) from 1 through to 3.

COPY Hold down the SHIFT key and use the arrow key to make a second cursor move up until it is on the first letter of a word or piece of program you wish to COPY. Release the SHIFT key and hold down the COPY key until the word or section is copied.

After you CAT a disc, use the arrow keys on their own to move the cursor to over the first letter of a title you wish to select. Hold the COPY key until the title (and the full stop BAS if you wish, but not the 1K or any more that is shown) has had the cursor run across it. Then press CONTROL, ARROW LEFT, ENTER.

Any words in all capital letters in the Tour of the Keyboard are Basic Language words that give commands to the computer, or Key names. You can not use a command word eg TIME as a variable, but 'timey' or 'itime' could be used as a variable. A variable can have any name or just a single letter.

AT LAST! Now you've looked at that, let's start by printing something on the screen and trying out the very basic functions of the computer.

The words in square brackets are instructions or explanations! Type in:

```
PRINT 3+3  [use lower case press ENTER]

 6         [answer shown on screen- try all examples.]
```

When you type in a BASIC KEYWORD in LOWER CASE in a numbered program and LIST it, the computer will change any KEYWORDS into UPPER CASE. Although all Keywords in this book are listed in Upper Case, they can be typed in in Lower Case. If a Keyword does not appear in Upper Case after it is LISTed, then you have made a spelling error or left out a space between two words.

```
PRINT 3*3      [press ENTER]

 9

PRINT 3-3  [ENTER]

 0

PRINT 3/3  [ENTER]
```

1

PRINT 3↑3 [ENTER]

27

PRINT 3+6-5 [ENTER]

4

PRINT 3+6-10 [ENTER]

-1 [Have you noticed that there has been a space left in front of the the answers? That is to allow space for a sign to be put there if it is needed. The computer leaves a space before and after a number unless it is between " " marks. It is called a 'leading' space.]

PRINT 3+6-5*2 [ENTER]

-1 [Has the computer made a mistake? No. Calculations are worked in the following order: MULTIPLY, DIVIDE, ADD and then SUBTRACT. The computer does not work from left to right as we do when working out a maths problem; to get the answer you expected, you need to use brackets. To remember the order ask yourself 'What must I do? I Must Do Another Sum!'- M,D,A,S - Multiply, Divide, Add, Subtract. The first letter of each word.

PRINT (3+6-5)*2 [ENTER]

8 [Now this works as you would expect it to because the calculations inside brackets are done first, even if there are brackets inside brackets.]

Try these examples on paper first and then do them on the computer to see if your calculations are the same.
Remember to press ENTER each time.

a.PRINT 12+5+3-18
b.PRINT 12+(5*3)-18
c.PRINT 2+5*3+7-3
d.PRINT 2+3+(5*4)*2
e.PRINT 2↑4
f.PRINT 18/6*3
g.PRINT (18/6)*3
h.PRINT 32-12*5/2+1

You didn't expect there to be no tricky ones, did you? Check how you went in the Answer section at the end of the book.
Did you peek? or were you careful and checked them through on the computer...

You can do a long line of adding and subtracting etc using a PRINT statement. The only limit is that you must stop before you exceed 255 characters in your PRINT statement eg:

PRINT 1.89+0.56+0.32+1.25*3+2.60+4.89 etc [RETURN]
(from a checkout docket.)

The zero before the decimal point is optional. For a bank statement you would start with the opening balance and subtract withdrawals and add deposits to get a current total- no dollar signs are necessary eg:

PRINT 347.00-12.50+245+32.65-109.05-198.85 [RETURN]

Note that the decimal point and two zeros is not necessary if it is a whole dollar. If your statement is too long, over 6 lines ie 255 characters, then a bell would sound at the 255th and no further characters could be added to that 'line' of statement. The length of a computer line is 255 characters. Hold down the letter 'a' now until the bell sounds and you will see how much room there is in a line of program.

RUN is another BASIC word you need to know. To give your brain (and mine) a much needed rest, take side 4 of your Utilities Discs and insert it in the disc drive and type:

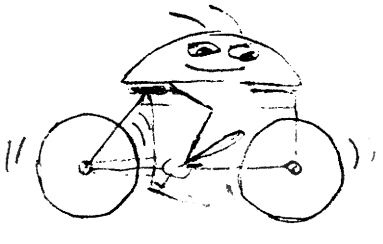
RUN"disc [RETURN] This will run a demonstration program.

Note:
CP/M discs }
Utilities discs} All names given to the discs that
Master discs } came with the computer.

BASIC is an easy language that is converted to Machine Language by the computer. Machine language is also called Machine Code.

[CONTROL,ARROW LEFT,ENTER] = RUN"

RETURN (the same as ENTER) = go to a new line, and/or carry out a command.



CHAPTER TWO REALLY MOVING..

Besides being able to interpret BASIC the computer can interpret CP/M - the language used on the CP/M Discs. If you wanted to find out what was in a book, you would look up the index. An index gives a catalogue or directory of subjects. As the two languages cannot use the same word and the word must not be too long, Basic shortened catalogue to CAT and CP/M shortened directory to DIR. CAT [RETURN] side one of the CP/M discs now. We are using BASIC at the moment, and we can CAT a CP/M disc; once we start CP/M, then we must use DIR to read its index.

Look for a program called DISCKIT3. This is a utility program that you use to FORMAT or COPY a disc. Take the CP/M disc out of the disc drive and compare it with a new disc. Notice that the new disc has a white lug or flap at the top near the corners? These are called the WRITE-PROTECT FLAPS. The CP/M Master Discs do not have them; you cannot accidently SAVE anything on those discs or ERASE from them. Make sure the flaps are up now on the new discs, so that that you can SAVE to them.

With Basic, when you start up you get the READY sign and the square CURSOR underneath; with CP/M, you get A> and the CURSOR alongside to prompt you to give the computer a command. READY and A> are called 'prompts'. The A means Disc Drive A.

FORMAT: This is the disc layout that is made by the computer by using the CP/M DISCKIT3 program and must be done to all new unused discs. The write-protect flap (the white bit) at the top of the disc must be up on the new disc. CAT the disc to be sure it is unused. The screen should read: Drive A: Read fail. Retry, Ignore or Cancel. You press C and get: Bad Command . The disc is unformatted. Insert Side One of the CP/M discs; hold down the SHIFT key and press the @ key (key No.26) for the 'bar'symbol:

:CPM [RETURN]

When you see A> type DISCKIT3 thus:

A>disckit3 [RETURN] [and follow the menu instructions.]

If you have not made working copies of your CP/M discs, do so now by pressing f7, the number 7 key on the right hand keypad-(more jargon!). This will format and copy at the same time-follow the instructions that appear on the screen. Do all 4 sides onto 4 separate sides of two new discs. Number and name the copies. Push down their write-protect flaps. Store the Master discs away safely. Now do a working disc.

To format a working disc, press f4 and select data format when the next menu appears, by pressing f6. Now insert the disc to be formatted and then press Y to confirm and formatting starts. You use the copy of the Master discs now to do formatting.

WARNING: Always CAT a disc before you format it - if you format a used disc you will remove all programs from the disc.

With a disc ready to use, information can be stored on to it and used from it by giving commands to the computer. When information has been SAVEd to a disc you have made a record of it- it stays there until you erase it. When you RUN the program later, it is not removed from the disc until you instruct the computer to do so.

Before we leave the CP/M discs for the moment, you use Side 1 when you want to enter CP/M. You must first type: |cpm and press RETURN. This is called 'booting' CP/M (jargon again!). Boot CP/M then take that disc out, insert Side 3 in the disc drive and type HELP, and you can read the HELP files. Type:

|CPM [You don't need me to say press RETURN now, each line.]

A>help [There is much to read there, but don't expect to understand it- this is just to tell you how to look at the Help files when you want to know what is on some of the Master Discs Unless you have used computers before you won't be ready to use CP/M for a while. This book is concentrating on BASIC.]

The computer understands a small number of words but really works only on numbers. There is a built in interpreter that translates Basic into numbers or rather Machine Language (also called Machine Code- two jargon words. The reason I am concentrating on numbers first is that you need to be able to send your messages to the computer in a set way, and unless you remember to think mathematically like the computer, you can make mistakes in your instructions. Some of my greatest errors have been my own fault- when I have been trying to write a program. I've worded things my way and not followed the laws of maths. Maths was my worst subject at school and it keeps turning up to haunt me! I wish someone had explained things as simply to me then as I shall to you in this book.

PRINT is the most widely used Basic command. To indicate why we are looking at certain numbers we need to make a heading and set things out neatly. PRINT will do both. Type:

PRINT "My Address" [remember to press ENTER]

My Address

Did you get the quotation marks right? You hold down the SHIFT key and press '2'. Try some more PRINT statements. Remember a PRINT statement must have quotation marks before and after the words you want to put on the screen.

To PRINT several things at once without the READY prompt coming in between each line, several groups of words can be put in the same command. To show that each command is separate, a colon is

used. That is why you often find lots of colons in the one line of computer instructions.

The number of characters per line in this book are more than you use on the screen of your computer. There are 64 characters to a line on this page, whereas in MODE 1, the usual MODE to use when typing in a program, there are 40 characters across the screen. With programs, I have printed 40 characters to a line in this book. A character on a new line here, therefore, will be in the same position on your screen. If it isn't, then you have missed a character. Do not press RETURN until all the line is typed in, even if it fills more than one row across the screen.

Type in, without stopping to press RETURN:

```
PRINT"My name is Judith Thamm":PRINT"I l
ive in Two Wells":PRINT"in P.O. Box 269"
                                    [now press RETURN]
My name is Judith Thamm
I live in Two Wells
in P.O. Box 269
```

Notice the colon between the speech marks and the word 'PRINT'. This puts the next words onto a new line.

A PRINT statement to the screen is important as it can be used as a heading or to give information on the screen. But in one long line like that example, a print statement can be confusing.

To make instructions easier to follow, and to give long instructions, we write them in a program. Numbering goes in tens so that if extra lines need to be added, the numbers in between can be used.

PRINT statements doing maths calculations do not need to have quotation marks, but headings and other words do. Here is a program to show this: [Press ENTER after each line.]

```
10 print "THIRTEEN TIMES TABLE"
20 print "13 x 0 = ";13*0
30 print "13 x 1 = ";13*1
40 print "13 x 2 = ";13*2
```

EXPLANATIONS:
10 The heading. Use Lower Case except where the words are between quotation marks.

20 to 40 Each line is the same. 13 x 0 = is to be PRINTed on screen, the semi-colon means, and-right-next-to-it put 13 multiplied by a number(the actual computer workings).

Now before this gets beyond my mathematical powers lets stop and look at what has been done.

Type LIST and press ENTER.

Some of the program has changed! Because PRINT is a BASIC LANGUAGE word, it is now in Upper Case (or capitals).
This is how the program will look now:

```
10 PRINT "THIRTEEN TIMES TABLE"
20 PRINT "13 * 0 = ";13*0
30 PRINT "13 * 1 = ";13*1
40 PRINT "13 * 2 = ";13*2
```

Type RUN and press ENTER.....
Did the program work or did you get an ERROR MESSAGE?

If you got an error message (SYNTAX ERROR) correct the error and then press the RETURN or ENTER key and RUN the program again. Use the arrow key to move the cursor along the line and put the cursor to the right of where you want to make the insertion or DELetion. You can delete and insert very easily. Use the DEL key for delete and just type in what you want to insert; no extra space is needed to add or insert letters or spaces. Put the cursor to the right of the position where you need to make an insertion, and type away. Hold the DEL key down and everything to the left of the cursor will be removed.

If you place the cursor over a letter to be removed, use the CLR key. Hold the CLR key down and everything under and to the right of the cursor will be removed.

Let us analyse the program so far; this helps to see if it is doing what you intended it to do:

10 Tells the computer to PRINT a heading in capital letters.

20 Tells the computer to PRINT 13 multiplied by 0 equals on the screen in the way we are used to seeing it written. Next to it (; means 'next to it' here), work out the answer. Note that the instruction must be given twice- once to go on screen (in quotation marks) and then again in computer language so that the answer is PRINTed too.

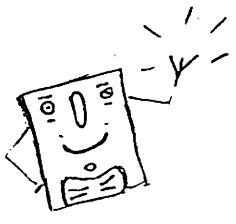
30 and 40 are the same as 20 except for the increase in the number that 13 is multiplied by.

This does seem a long way to work something out! Is it going to be so difficult to do any small problem? Of course not. Now you understand how things must be done, write a small program based on the first one.

In the first line PRINT the title My Disc Box. (Line No. 10)
In the second line PRINT that you have Blank Discs, 2 boxes of 10 and let us imagine that you can't work out 2*10 and need to put it in the PRINT statement.
In the third line PRINT how many CP/M Discs you have: 2 and don't for get the 2 copies you made. Show and PRINT workings.

Now RUN the program.

Here is how your program should have looked:



```
10 PRINT "My Disc Box."
20 PRINT "Blank Discs : 10*2=";10*2
30 PRINT "CP/M Discs : 2*2=";2*2
```

and when you RUN it:

```
My Disc Box.
Blank Discs : 10*2=20
CP/M Discs : 2*2=4
```

If you got that correct you can proceed. If not go back and try it again as we are going to do something more involved next.

Any time you want to perform the same task more than once, you you can tell the computer to do it for you- FOR being the key word. Basic words are simple. This next program goes a step further in writing out a table.

LISTING for 13X10 [USE LOWER CASE TO TYPE IN.]

```
10 PRINT "THIRTEEN TIMES TABLE"
20 FOR number = 0 TO 5
30 PRINT "13 x" number "="; 13*number
40 NEXT number
```

EXPLANATIONS:
10 The heading, between quotation marks, must be in upper case.

20 If you miss the spaces between: 0 TO 5 you will get an error message. Decides (for) how many times we want the task performed, in this case 6. Multiply 13 by 0, then 1,2,3,4,5- six times. The computer counts from 0 as its first number. The word 'number' is being used as a VARIABLE. Each time 13 is multiplied by 'number', 'number' has a different value. That is why it is called a variable, because it varies or changes.

30 Print 13 times multiply(x) and then the number. Print the equals sign right next to them; put the answer of the workings, 13 multiplied by (using the * sign) what the number has become.

40 Go back and do the next number until all are done. This sends the computer around and around those few lines until all six repeats are completed. This is called a FOR....NEXT loop.

First LIST the program.

Correct any errors and then RUN the program. This should be the result.

```
THIRTEEN TIMES TABLE
 13 x 0=0
```

```
13 x 1=13
13 x 2=26
13 x 3=39
13 x 4=52
13 x 5=65
```

The FOR ... NEXT combination is called a LOOP because it sends the computer around to do the same thing again a specified number of times.

Try changing the number (if you are superstitious) and then alter line 20 so that the table runs through the loop 11 times.

To Change a line in the program, type:

EDIT 20 [Line 20 will appear with the cursor ready for you to move it using the arrow keys. DEL the 5 and insert .... ?]

Go back to the explanation of line 20 above, did you put 10 or 11 ? It should have been... Begin counting from 0... the lower number of course! Line 20 should read:

```
20 FOR number = 0 TO 10
```

LIST the program and then RUN it. If you put 11 then the program would run through the loop 12 times.

This time, once you have got this program running correctly, SAVE it. Insert a blank formatted disc in the disc drive and type:

```
SAVE"13X10          [Press RETURN]
```

The name on the disc catalogue can be up to 8 letters (or numbers and the signs '&' and hyphen) in length, before the full stop which the computer will add, along with the words BAS or BAK.

NOTE: This program (and all others) is needed again later. Each program is used again to build another program.

CAT the disc to see if the program has been recorded. You should see:

```
DRIVE A: user 0

 13X10    .BAS    1K

 177K  FREE
```

Now supposing you wanted to PRINT more than one set of tables on the screen at once , how could you do it? Here is the first of some ways to do it.

Spaces are made inside the PRINT statement to set things out:

LISTING for TABLES:

```
10 PRINT "                    TABLES"
20 FOR n=0 TO 10
30 PRINT "13 x" n"=";13*n"          14 x"
n"=";14*n
40 NEXT n
```

EXPLANATIONS:
10 Press space bar 17 times before TABLES".

20 FOR starts the loop. 'n' is short for number (the variable). 0 TO 10 means to go around 11 times - 0,1,2,3,4,5,6,7,8,9,10 = 11 times.

30 PRINT out 13 multiplied by 'n', put in the equals sign, give the answer, then leave 8 spaces, and PRINT out 14 multiplied by 'n' etc.

40 NEXT sends the computer back to do another loop until it has done the number in the FOR instruction. The n is put there to tell which loop is to be done (necessary when more than one loop is used in a program).

LIST the program and correct any errors.

RUN the program. It should look like this:

```
                TABLES
13 x 0 = 0          14 x 0 = 0
13 x 1 = 13         14 x 1 = 14
13 x 2 = 26         14 x 2 = 28
13 x 3 = 39         14 x 3 = 42
13 x 4 = 52         14 x 4 = 56
13 x 5 = 65         14 x 5 = 70
13 x 6 = 78         14 x 6 = 84
13 x 7 = 91         14 x 7 = 98
13 x 8 = 104         14 x 8 = 112
13 x 9 = 117         14 x 9 = 126
13 x 10 = 130         14 x 10 = 140
```

Not very tidy to look at, but did yours work out? Find the error or errors, correct them, and RUN the program again, then:

SAVE"TABLES

If you have to stop now the program will be there for you to retrieve and use again. To check if the program SAVEd to disc, type CAT and see if the title is on the disc:
TABLES .BAS 1K

In computer jargon the title of something you SAVE is a Filename, made up from 1 to 8 characters, with the limitations

mentioned before, then a full stop and 3 more possible letters. Usually the last 3 are left out because the computer will put ..BAS showing it is a BASIC file, or ..BIN for a Binary (machine code) file. I use them sometimes to show where I got a program from eg .TXT to show it came out of the Manual.

The 1K shows how much disc space has been used.

If you SAVE a program with exactly the same name again, ie TABLES, this time the newest program will appear on the disc catalogue as TABLES .BAS and the old program will have its name changed to TABLES.BAK (meaning back-up). The third time you save a program by exactly the same name, the first program is erased, the second has its name changed to TABLES.BAK and the newest program becomes TABLES.BAS.

If you have a program written ready to save, and you think you have used a chosen name before, while the program has not yet been SAVEd, you can CAT the disc, erase or rename a file, and then SAVE the program, without losing or harming the original program. You can not reset the computer or RUN another program with the same line numbers without losing the first one.

WHAT HAPPENS BY SAVING THE SAME NAME THREE OR MORE TIMES:

1st time:
TABLES becomes TABLES.BAS [1] the number is to help identify it.

2nd time:
TABLES becomes TABLES.BAS [2] and TABLES.BAS [1]
becomes TABLES.BAK [1] it is still the first one SAVEd.

3rd time:
TABLES becomes TABLES.BAS [3] and TABLES.BAS [2]
becomes TABLES.BAK [2] and TABLES.BAK [1] is erased.

If you decide that TABLES.BAK [2] is the program you want rather than the TABLES.BAS [3] and erase TABLES.BAS, you will be left with only TABLES.BAK on your disc as a title [No 2]. You need to change its name to a ....BAS title unless you remember to add the stop and BAK each time you try to run the program.

To change a name of a file on a disc, use |REN (called Bar REN for RENname).

```
|REN,"newname.bas","oldname.bak"
                                ie |REN,"TABLES.BAS","TABLES.BAK"
```

[You can type in titles in lower case.]

CHAPTER THREE USING THE INDICATORS...

Another way to set out is to use the COMMA in a print statement. LIST the program again, or if you had to stop, LOAD"TABLES or RUN"TABLES and then press the ESC key twice and LIST the program.

SAVE each tables program as you do them, as we will be using each one again in another program later.

Change a line in the program. Type:

EDIT 30 [RETURN]

```
30 PRINT "13 x" n"=";13*n"          14 x"
 n"=";14*n
```

insert a comma after 13*n, and DELete the spaces before 14:

```
30 PRINT "13 x" n"=";13*n,"14 x" n"=";14
*n
```

RUN the program again.

```
 13 x 0 = 0          14 x 0 = 0
 13 x 1 = 13         14 x 1 = 0
                        etc
 13 x 8 = 104                    14 x
  8 = 112                                 etc
```

Notice the difference? Much tidier to start with, but why has the table jumped all over the screen after 7 times..? and gone on to the next line?

The comma is used to make ZONES. The default setting (the one that is there when the machine is first switched on) for a ZONE is 13 characters.

This is where the leading space before and after a number affects the layout.

To illustrate this, do not use a line number-

```
PRINT 3"plus"5"are"8;"3plus5are8"              [RETURN]

 3 plus 5 are 8 3plus5are8
```

Notice the space left before and after the numbers. When the last column of the 13 times 8 was printed the final space after the last number then over ran into the next ZONE which then caused the computer to jump across to the third ZONE for the 14 times 8.

The default width of the ZONE can be changed by adding a ZONE

command to the program. List the program and add this line:

```
15 ZONE 16
```

and RUN the program. Now the table will be neatly set out.

```
SAVE"TABLEZON
```

Another way to make layout neat and tidy is to use the TAB command. First remove line 15. Type:

```
15                and press RETURN.
```

This deletes a single line. If you do this to a line you needed you have to hope that the program was already saved to disc!

Now EDIT line 30 to read :

```
30 PRINT TAB(5) "13 x" n"=";13*n;TAB(25)
 "14 x" n"=";14*n                                [NB the extra ;]
```

and run the new program.

```
SAVE"TABLETAB
```

TAB works much the same way as on the typewriter. It allows you to make a neat layout when you write a program. Each number in the brackets after the TAB command refers to a COLUMN on the screen. The default setting of the screen has 25 ROWS down and 40 COLUMNS across. The TAB command sends the next character to the column numbered in the brackets. TAB(10) means column 10.

Lastly, list the program again and edit line 10:

```
10 PRINT SPC(17) "TABLES"                       [change the wording]
```

and RUN the program. SPC is short for spaces and the number in the brackets starts from the left. These are all the PRINT commands you need to know to send most items to the screen.

```
SAVE"TABLESPC
```

Notice that we have to send one command to the computer to operate and another command for us to be able to see what is going on. If we want to operate a printer as well, we must send a third command. If you do not have a printer as yet, it will still be useful to know what to do when you do get one.

LIST line 30 thus: LIST 30 Start a new line called 35 and then add #8, ... and COPY the rest of the line to read:

```
35 PRINT #8, TAB(5) "13 x" n"=";13*n;TAB
(25) "14 x" n"=";14*n
```

and no other changes. Connect up the printer cable and plug, switch on the printer, insert paper, and run the program.

[You must leave line 30 in as well as adding line 35.]

If you don't have a printer yet, REM line 35 (Make it a comment line, not a command line).

EDIT 35 to read:

35 REM PRINT #8, TAB(5) "etc [the rest is unchanged]

When you want to use a printer, just remove the REM and the program will print out for you.

SAVE"PRINTABL

To show how important it is to get things in the right order,

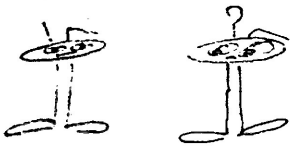
LOAD"13X10

LIST it and add the following line:

35 number = number + 1

Now run the program. Can you explain why you get this result?

13 x 0 = 0
13 x 2 = 26
13 x 4 = 52
13 x 6 = 78
13 x 8 = 104
13 x 10 = 130



The value of number is zero, then zero has one added to it in line 35. That was the first loop. One is added by the loop count, so now the value of number is 2. Each time in fact, 2 is added. Test line 35, add a REM to it:

35 REM number = number + 1

and RUN the program again to check that it was right at first.

After you have run it LIST line 35. Make a line 25 and COPY the contents of 35, except for the REM.

25 number = number + 1 RUN the program again.

This time you will get the odd numbers. The value of number starts at zero, then line 25 adds on one and line 40 sends the loop back to be run again. One more is added on for the loop count, and then line 25 adds on one, and so once again the program runs in steps of two. Where you place a line often

alters a program dramatically.

Add line 5 and then SAVE"13SWITCH

```
5 REM Switch REM in line 35 to line 25
```

SAVE"13SWITCH

Finally here is a way of counting or working consecutive stages.

LIST the program again. Delete line 25 by typing just:

25 [RETURN]

then EDIT line 20

```
20 FOR number = 1 TO 10 STEP 2
```

RUN the program and you should get:

```
13 x 0 = 0
13 x 2 =26 etc. advancing in stages of 2 times 13.
```

The program goes forward in STEPs of two with each increase of the loop count. This is often used in graphics, but not just STEPs of two. Try a STEP of 5.

SAVE"13STEP

Now you have several short example programs taking up a lot of disc space. The most logical step next, is to create a Menu and put all these programs into it, so that they are in a group and take up less Disc Space.

CHAPTER FOUR TAKE OUT YOUR ROAD MAP...

A Menu, in computer jargon, is an index that allows you to use files that have been grouped together in a 'book' of programs. Sometimes a Menu is just a list of titles, as in a Disc Menu. This gives ease of selection either by moving an arrow to the title or by choosing a number. But when another program is required, the Disc Menu must be run again, or CAT used to read the titles in the normal way.

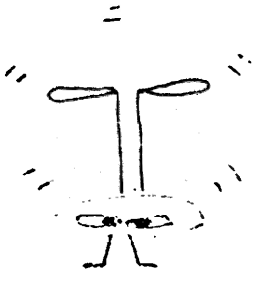
A Menu that controls a group of programs will show only one title for all the files that are included and until the program is run, the titles will not be seen. Such a menu is very useful because it will save title space and program space on a disc.

There is only room for 64 titles in the Disc Directory. If you have used 64K in programs 1K in length, then you cannot add more to the disc despite there being 104K FREE because the Disc Directory would be full.

Many programs that are only 1K in length are extremely useful for reference when you first begin. If you place such programs in a Menu, they can have a longer title that better indicates what the program is about. A program can have a number of REM statements at the beginning that you can LIST and read. The REMark statements can contain explanations, instructions and perhaps hints on how to use a program.

LISTING for MENUPLAN: [Read EXPLANATIONS before you type in.]

```
10 REM add small programs to MENUPLAN st
arting from line 30.
20 GOSUB 5000: REM setting up
5000 REM --- set up screen ---
5010 MODE 1:CLS
5020 INK 0,11
5030 INK 1,0
5040 BORDER 17
5050 WHILE NOT done
5060 PRINT TAB(16) "MENUPLAN"
5070 PRINT TAB(16) "---------"
5080 PRINT:PRINT TAB(12) "1.    FIRST"
5090 PRINT:PRINT TAB(12) "2.    SECOND"
5100 PRINT:PRINT TAB(12) "3.    THIRD"
5110 PRINT:PRINT TAB(12) "4.    "
5120 PRINT:PRINT TAB(12) "5.    "
5130 PRINT:PRINT TAB(12) "6.    "
5140 PRINT:PRINT TAB(12) "7.    "
5150 PRINT:PRINT TAB(12) "8.    "
5160 PRINT:PRINT TAB(12) "9.    END PROGR
AM"
5170 LOCATE 12,23:PRINT "Select a number
..."
5180 PRINT:PRINT TAB(6) "Press a key to
```



```
return to menu."
5190 k$=INKEY$:IF k$="" THEN 5180
5200 ON INSTR("123456789", k$) GOSUB 30,
100,200,300,400,500,600,700,5220
5210 WEND
5220 done=0
5230 CLS:LOCATE 10,12:PRINT "Program Con
cluded."
5240 END
```

Use LOWER CASE (small letters) while typing in the program, except for words in double speech marks. The computer changes command words into UPPER CASE (capitals).

For this first longer program, I have put every line above its explanation. The line is in LOWER CASE exactly as you will see it typed onto your screen before it is LISTed. After you LIST the program, it should look like the PROGRAM LISTING above. There are many new terms to explain too.

EXPLANATIONS [and program lines as they will look when first typed in.]

```
10 rem add small programs to MENUPLAN st
arting from line 30.
```
-- The REMark reminds you that the first program will be renumbered from 30.

```
20 gosub 5000: rem setting up
```
-- GOSUB sends the computer directly to the number indicated, in this case to set up the screen- a program can have many REMarks.

Now type: AUTO 5000 Press RETURN. Automatic numbering in 10's will start from 5000. You do not need to enter the line number until you press ESC to stop AUTO. One press of ESC pauses, two presses interrupts or stops. Press a key to continue after a pause.

```
5000 rem --- set up screen ---
```
-- Another REMark line, to show where a section begins.

```
5010 mode 1:cls
```
-- Sets the number and size of characters, and the number of colours that the screen will allow. MODE 1 allows 40 characters across in 4 colours. The colon shows the end of one instruction and the start of another. CLS gives a clean start to a program.

```
5020 ink 0,11
```
-- The PAPER or background colour. INK 0,.. is always the background, and the second number is the number of the colour from the colour chart.

```
5030  ink 1,0
```
-- INK 1,... is always the writing colour, and the second number is the number of the colour from the colour chart.

```
5040 border 17
```
-- Sets the BORDER colour, again from the colour chart.

```
5050 while not done
```
-- This is the start of another kind of loop. A WHILE...WEND looks for a condition to occur before it will stop whereas a FOR...NEXT loop counts how many times it runs. In this case the WHILE ... WEND loop will keep running until 'done' happens.

```
5060 print tab(16) "MENUPLAN"
```
-- Something familiar at last! PRINT the heading at column 16.

```
5070 print tab(16) "--------"
```
-- Hold down SHIFT, use the arrow key and COPY up to the first double quotation marks. Use the hyphen key to underline. Don't miss out the second set of quotation marks.

```
5080 print:print tab(12) "1.   FIRST"
```
-- The first PRINT gives a blank line, because nothing is listed to be printed. The second PRINT moves on to the next line. The first title is to be PRINTed at column 12. There are 3 spaces after the number 1.

```
5090 print:print tab(12) "2.   SECOND"
5100 print:print tab(12) "3.   THIRD"
5110 print:print tab(12) "4.   "
5120 print:print tab(12) "5.   "
5130 print:print tab(12) "6.   "
5140 print:print tab(12) "7.   "
5150 print:print tab(12) "8.   "
5160 print:print tab(12) "9.   END PROGR
AM"
```

5090 - 5160 The same as 5080. Use the copy key. COPY up to and including 1 - then DEL 1 and insert 2 - COPY the full stop and 3 spaces then type in SECOND". Press RETURN and COPY up to line 5160 the same way.

```
5170 locate 12,23:print "Select a number
..."
```
-- LOCATE finds a character position in column 12 on the 23rd row (3 above the bottom of the screen). LOCATE works from the top left hand corner of the screen. If a WINDOW is used LOCATE works from the top left hand corner of a WINDOW.

```
5180 print:print tab(6) "Press a key to
return to menu."
```
-- PRINT a line with nothing, then on the next line at column 6 PRINT an instruction on the bottom line of the screen. There is a space after '..key to '.

```
5190 k$=inkey$:if k$="" then 5190
```
-- This line is a loop waiting for a key to be pressed. This (and slight variations) is a line you will meet and use often.

The k$ (or k dollar) is used as a STRING VARIABLE. A string variable can be letters or numbers or a mixture of the two.

The choice of letter for this line is usually between 'k' and 'a' because of association; the line is Asking for a Key to be pressed. So it's a matter of what sticks in your mind which you choose to use later. A complete word can be used. It is easier when you are trying to write a program if you get into the habit of using the same string variables all the time, then you won't confuse yourself and you can COPY them- less keying in.

INKEY$ is the inquiry string [inkey- 'inq'uiry, sound-a-likes]. 'k' is going to be the name given to the inquiry string. The inquiry string asks if a key has been pressed.

If a key has not been pressed ie if 'k' string has nothing in between the two lots of quotation marks - k$="" - (NB no spaces between the ""), then go back to the beginning of line 5180 again and again until a key is pressed.

```
5200 on instr("123456789",k$) gosub 30,
100,200,300,400,500,600,700,5220
```
-- ON INSTR(" ", $) GOSUB- you can see the punctuation etc that belongs to the command clearly now. When the numbers listed between quotation marks are pressed, [on instruction 123 etc], go to the sub-routine [GOSUB] at line 30 for number 1, at line 100 for number 2, at line 200 for number 3, and so on. The line numbers must be entered in the same sequence as the numbers in the quotation marks. ON INSTR(" ", $) GOSUB is a special line that needs a key pressed before the program continues. Any line number can be called on to be a GOSUB line if it is the line the computer needs to be sent to. 'k$' will be a number from 1-9.

```
5210 wend
```
-- The end of the WHILE ... WEND loop.

```
5220 done=0
```
-- The condition needed to stop the WHILE ... WEND loop.

```
5230 cls:locate 10,12:print "Program Con
cluded."
```
-- Clear the screen, find the character position 10 columns in and 12 rows down and PRINT...

```
5240 end
```
-- Another simple BASIC word that means what it says.

LIST the program and check it against the LISTING for MENUPLAN. RUN the program. Press a number key between 1 and 8. You should get the Error Message: Line does not exist in 5200. RUN again and press 9. The screen should clear and 'Program Concluded' should be printed in the centre of the screen, with the Ready sign at the side. The program is then ready to use.

```
SAVE"MENUPLAN
```

CHAPTER FIVE PARKING...

Placing the programs that we have done into the MENUPLAN program is fairly simple. None of the programs have graphics or other complicated material in them. You need the disc with all the programs so far on it in the disc drive.

LOAD" or RUN" 13X10 and LIST it. Then type:

```
5 REM --- 13x10 ---
RENUM 30,5,10
```

[Start at 30, renumber from line 5, counting in 10's.]

LIST the program again. The numbering should now start from 30. Leave the disc in the disc drive ready to continue.

Add these lines:

```
80 LOCATE 12,25:PRINT "Press a key..." [On screen instructions]
90 k$=INKEY$:IF k$="" THEN 100          [Wait for a key press]
100 GOTO 5000                           [Sends back to Menu]
```

MERGE "MENUPLAN

LIST the program and press ESC to pause and see that lines 30 to 100 are there.

LIST 10-100 [List from line 10 to line 100.]

```
10 REM menuplan...      [etc.]
20 GOSUB 5000
30 REM --- 13x10 ---
40 PRINT "THIRTEEN TIMES TABLE"
50 FOR number = 0 to 10
60 PRINT "13 x" number "="; 13*number
70 NEXT number
80 LOCATE 12,25:PRINT "Press a key..."
90 k$=INKEY$:IF k$="" THEN 100
100 GOTO 5000
```

EDIT 10

```
10 REM a group of programs showing how t
o do setting out.
```

Test the program by RUNning it. Select Number 1.

Oh dear...one program has run over another on the screen.
To overcome this, EDIT line 40.

```
40 CLS:PRINT "THIRTEEN TIMES TABLE"
```

Insert CLS: at the beginning of the line, at the cursor. Don't

miss out the colon. Each program will need CLS added.

Try RUNning the program again. Select 1, the 13 times table should be printed, along with the Press a key instruction at the bottom of the screen.
Press a key. If the program returns to the menu, it's working.

As all the programs are basically about printing tables to the screen, the name of the Menu could become BASIC SETTING OUT. Set the computer to:

AUTO 5060 and we'll change the title line and all the other lines in that section at the same time.

5060 Change TAB(16) to TAB(12), delete MENUPLAN and insert the new heading. Put the cursor on the last quotation mark and press DEL to delete the word MENUPLAN.

Use CAPS LOCK and type in:

```
5060 PRINT TAB(12) "BASIC SETTING OUT"          [Press RETURN]
```

5070 Change the TAB number to 12, and add more hyphens between the double quotation marks to match the new heading.

```
5080 PRINT:PRINT TAB(8) "1.   THIRTEEN T
IMES TABLE"
```

5080. DELete the word FIRST and insert THIRTEEN TIMES TABLE in its place.

Continue in the same way with the rest of the titles.

```
5090 PRINT:PRINT TAB(8) "2.   13 TIMES &
 14 TIMES"
5100 PRINT:PRINT TAB(8) "3.   TABLES USI
NG ZONES"
5110 PRINT:PRINT TAB(8) "4.   TABLES USI
NG TAB"
5120 PRINT:PRINT TAB(8) "5.   TABLES USI
NG SPC"
5130 PRINT:PRINT TAB(8) "6.   TABLES USI
NG A PRINTER"
5140 PRINT:PRINT TAB(8) "7.   TABLES ODD
S & EVENS"
5150 PRINT:PRINT TAB(8) "8.   TABLES USI
NG STEP"
                                  [Change TAB to 8 in 5160]
5200 ON INSTR("123456789", k$) GOSUB 30,
110,200,300,400,500,600,700,5220
```

5200 Put in the only line number change , 100 becomes 110.

RUN the program and see that you have done the alterations correctly.

SAVE"SETOUT [Most important.]

To add the next program, follow the same procedures:

1. LOAD the next program.
2. Use a REM to show the filename, and to make the new section stand out. The line number is 5 each time.
3. Insert CLS: at the beginning of line 10.
4. RENUM. [For the next program, RENUM 110,5]
5. MERGE"SETOUT
6. LIST 80-100 These 3 lines are ready now for you to COPY on to the end of the second program. The new line numbers will be 160 to 180. In other words line 160 will be a copy of 80, line 170 will be a copy of 90 and line 180 will be a copy of 100. Copy the same three lines at the end of each program. Don't forget the line number at the end of line 90 must change each time it is copied.
7. SAVE"SETOUT

By LISTing a line in order to COPY it, you are not moving that line out of place, you have just borrowed it to look at it; it will still be in its original place in the program.

Repeat the seven steps for each of the other short programs.

TABLEZON will RENUM from line 200.
TABLETAB " " " " 300
TABLESPC " " " " 400
PRINTABL " " " " 500
13SWITCH " " " " 600
13STEP " " " " 700

SAVE"SETOUT after you have RENUMbered and MERGEd each section.

When you have completed SETOUT, and tested that each section works properly, CAT the disc and compare the space taken up by SETOUT to that used by the eight other programs- nine if you include MENUPLAN- quite a difference!

THIS SECTION IS MOST IMPORTANT. MOVING PARTS OF A PROGRAM INTO A MENU IS THE BASIS OF ALL PROGRAMS. The 'menu' is actually the equivalent of a program CONTROL section.
Try to MERGE the whole program without checking the final listing given below.

LISTING for SETOUT:

```
10 REM a group of programs showing how t
o do setting out.
20 GOSUB 5000: REM setting up
30 REM --- 13x10 ---
40 CLS:PRINT "THIRTEEN TIMES TABLE"
50 FOR number = 0 TO 10
```

```
60 PRINT "13 x" number "="; 13*number
70 NEXT number
80 LOCATE 12,25:PRINT "Press a key..."
90 k$=INKEY$:IF k$="" THEN 90
100 GOTO 5000
110 REM --- tables ---
120 CLS:PRINT "                    TABLES"
130 FOR n=0 TO 10
140 PRINT "13 x" n"=";13*n"         14 x"
 n"=";14*n
150 NEXT n
160 LOCATE 12,25:PRINT "Press a key..."
170 k$=INKEY$:IF k$="" THEN 170
180 GOTO 5000
200 REM --- tables using zone ---
210 CLS:PRINT "                    TABLES"
220 ZONE 16
230 FOR n=0 TO 10
240 PRINT "13 x" n"=";13*n,"          14
x" n"=";14*n
250 NEXT n
260 LOCATE 12,25:PRINT "Press a key..."
270 k$=INKEY$:IF k$="" THEN 270
280 GOTO 5000
300 REM --- tables using tab ---
310 CLS:PRINT "                    TABLES"
320 FOR n=0 TO 10
330 PRINT TAB(5) "13 x" n"=";13*n;TAB(25
) "14 x" n"=";14*n
340 NEXT n
350 LOCATE 12,25:PRINT "Press a key..."
360 k$=INKEY$:IF k$="" THEN 360
370 GOTO 5000
400 REM --- tables using spc ---
410 CLS:PRINT SPC(17) "TABLES"
420 FOR n=0 TO 10
430 PRINT TAB(5) "13 x" n"=";13*n;TAB(25
) "14 x" n"=";14*n
440 NEXT n
450 LOCATE 12,25:PRINT "Press a key..."
460 k$=INKEY$:IF k$="" THEN 460
470 GOTO 5000
500 REM --- tables using a printer ---
510 CLS:PRINT SPC(17) "TABLES"
520 FOR n=0 TO 10
530 PRINT TAB(5) "13 x" n"=";13*n;TAB(25
) "14 x" n"=";14*n
540 PRINT #8, TAB(5) "13 x" n"=";13*n;TA
B(25) "14 x" n"=";14*n
550 NEXT n
560 LOCATE 12,25:PRINT "Press a key..."
570 k$=INKEY$:IF k$="" THEN 570
580 GOTO 5000
600 REM --- Switch the REM in line 630 t
o line 650 for evens
```

```
610 CLS:PRINT "THIRTEEN TIMES TABLE"
620 FOR number = 0 TO 10
630 number = number + 1
640 PRINT "13 x" number "="; 13*number
650 REM number = number + 1
660 NEXT number
670 LOCATE 12,25:PRINT "Press a key..."
680 k$=INKEY$:IF k$="" THEN 670
690 GOTO 5000
700 REM --- tables using step ---
710 CLS:PRINT "THIRTEEN TIMES TABLE"
720 FOR number = 0 TO 10 STEP 2
730 PRINT "13 x" number "="; 13*number
740 NEXT number
750 LOCATE 12,25:PRINT "Press a key..."
760 k$=INKEY$:IF k$="" THEN 760
770 GOTO 5000
5000 REM --- set up screen ---
5010 MODE 1:CLS
5020 INK 0,11
5030 INK 1,0
5040 BORDER 17
5050 WHILE NOT done
5060 PRINT TAB(12) "BASIC SETTING OUT"
5070 PRINT TAB(12) "-----------------"
5080 PRINT:PRINT TAB(8) "1.    THIRTEEN T
IMES TABLE"
5090 PRINT:PRINT TAB(8) "2.    13 TIMES &
 14 TIMES"
5100 PRINT:PRINT TAB(8) "3.    TABLES USI
NG ZONES"
5110 PRINT:PRINT TAB(8) "4.    TABLES USI
NG TAB"
5120 PRINT:PRINT TAB(8) "5.    TABLES USI
NG SPC"
5130 PRINT:PRINT TAB(8) "6.    TABLES USI
NG A PRINTER"
5140 PRINT:PRINT TAB(8) "7.    TABLES ODD
S & EVENS"
5150 PRINT:PRINT TAB(8) "8.    TABLES USI
NG STEP"
5160 PRINT:PRINT TAB(8) "9.    END PROGRA
M"
5170 LOCATE 12,23:PRINT "Select a number
..."
5180 PRINT:PRINT TAB(6) "Press a key to
return to menu."
5190 k$=INKEY$:IF k$="" THEN 5190
5200 ON INSTR("123456789", k$) GOSUB 30,
110,200,300,400,500,600,700,5220
5210 WEND
5220 done=0
5230 CLS:LOCATE 10,12:PRINT "Program Con
cluded."
5240 END
```

CHAPTER SIX CHECKING THE LIGHTS

What else can the computer do? As I am forever mentioning characters, let's have a look at the second set of characters in the keyboard.

LISTING for CHARSET:

```
10 REM Graphics characters and modes
20 INK 0,15:INK 1,0:m=2
30 GOSUB 170
40 GOSUB 180
50 GOSUB 190
60 GOSUB 200
70 PRINT " GRAPHICS CHARACTERS"
80 PRINT
90 FOR n=126 TO 255
100 PRINT CHR$(n);
110 PRINT CHR$(128)+CHR$(128);
120 NEXT n
130 LOCATE 1,24:PRINT "Press a key. MODE"
;m
140 k$=INKEY$:IF k$="" THEN 140
150 m=m-1
160 RETURN
170 MODE 2:GOTO 70
180 MODE 1:GOTO 70
190 MODE 0:GOTO 70
200 MODE 1:END
```

EXPLANATIONS:

A MODE is a screen layout. MODE 1 is the default screen.
MODE 2 has 80 columns, 25 rows, allows 2 colours plus border.
MODE 1 has 40 " 25 " " 4 " " " .
MODE 0 has 20 " 25 " " 16 " " " .
Columns are counted across the screen from left to right: rows are counted down from the top of the screen.

10 REMark line.

20 The first number in an INK command calls the PEN of that number. INK 0,15 means get PEN 0 and fill it with colour 15. If the 15 was left out and the command was just INK 0, -then the colour would be 1 or blue in all MODEs. Lower case m is a variable that is used to set the number of the MODE. It is given the value of 2.

30 to 60 GO to the SUBroutine lines in sequence so that the MODE changes and the program is repeated- a CONTROL just like a menu!

70 The heading for the top of the screen.

80 PRINT alone means leave a blank row.

90 A counting loop. 'n' is short for number. The numbers used are the numbers of the Graphics Character Set. Turn to Page 13 in Chapter 7 of the User Instructions (referred to as the Manual in future) and follow the numbers under the symbol charts.

100 PRINT the character strings [CHR$] according to the number in the brackets. As n starts at 126, the first character to be printed will be number 126. The semi-colon means, put the next item right beside this. [Below 126 are alphabet etc.]

110 If you have your Manual open you will see that number 128 is a blank. A blank plus another blank makes two blank spaces to go right beside each character. This is to spread them out so you can see them more easily.

120 The send-back line in the loop.

130 Find column 1, row 24 and PRINT the key press instructions, then the MODE, and then its number.

140 The waiting-for-a-key-to-be-pressed line.

150 Take one away from the number of the MODE. [MODE 2 will become MODE 1]

160 Go back and do the next GOSUB.

170 Direct command to change MODE. The GOTO (line) 70 is the start of another kind of loop. Lines 180 and 190 are the same commands for other MODEs. GOTO is an endless loop.

200 Goes back to the default MODE and ends the program. [READY appears.]

LIST the program and check it. SAVE THE PROGRAM NOW BEFORE YOU RUN IT.

Programs in MODE 0 can wipe themselves off Memory when they are run, so SAVE before you RUN !!!

```
SAVE"CHARSET
```

RUN the program and correct any errors. Follow this system. Type:

```
MODE 1    [RETURN]
LIST      [RETURN]
```

If no INK colour should appear type:

```
INK 0,1:INK 1,0   [RETURN]
```

This last line can be used at any time when the writing INK and the background ink become the same. The first ink command turns the screen blue and the second writes to the screen with black

ink. If that still doesn't bring visible printing to the screen try :

```
MODE 2      [The program listing won't be identical to this book
             then, change back to MODE 1 again for that.]
```

Commands given without a line number do not upset a program that is currently in memory. If it has no line number, once a command has been carried out, it is forgotten, so to speak.

Every program in this book is used to develop another. So you need to SAVE each one.

Two commands that find positions on the screen are LOCATE and ORIGIN. So far we have used LOCATE a number of times. LOCATE is used to find a position on the screen in order to PRINT something.

The parameters (specifications in non-jargon language) of LOCATE are:

1. It can have a WINDOW command attached to it, followed by a comma. [LOCATE #1,24,12]
2. The first number is the column number (across), followed by a comma.
3. The second number is the row number (down).
4. The numbering starts from the top left hand corner of the WINDOW.

```
LOCATE #1,1,1          [Find WINDOW No 1, column No 1, row No 1.]
```

The parameters of WINDOW are controlled in the same way as LOCATE, except that four positions must be given to mark out all four edges of a WINDOW.

1. The default WINDOW is #0, . A #0, command is not usually used.
2. A WINDOW's size depends on the MODE.
3. The numbering goes:left edge,right edge,top edge,bottom edge.

```
WINDOW 1,40,1,25
```

[Default WINDOW MODE 1- the one when you switch on. Definition of DEFAULT: Default is de computer works dat way..]

PLOT is used in Graphics to find a position from which to draw. Unless an ORIGIN is given as a starting point for PLOT to work from the results can be strange.

ORIGIN works from the bottom left hand corner of the screen using x and y as coordinates. X measures across and Y measures up. Regardless of MODE, there are always 0 to 640 pixels across and 0 to 400 pixels up in the screen.
ORIGIN can be set to eg the centre of the screen. If another ORIGIN command is needed, you must work from the last ORIGIN

point. (Then minus left & down pixel counts are used.) ORIGIN moves to an invisible position. Use PLOT to show where that position is.

```
ORIGIN 320,200:PLOT 0,0,1
```

[Move to centre screen. PLOT zero pixels across and zero pixels up, using No 1 PEN.]

Try that now. A dot should appear in the centre of the screen. The same command with a final zero: PLOT 0,0, 0 would 'turn off' the visible PLOT.]

ORIGIN can set the position of a Graphics Window.

```
ORIGIN 0,0,0,640,400,0
```

is the default Graphics Window.

[Starting bottom left,at zero across, and zero up, mark a GRAPHICS WINDOW from zero left, to 640 right, by 400 at the top, and zero at the bottom.]

PLOT can be used invisibly if only the x and y coordinates are used. [The measurement across or up. If a minus is used, the measurement is to be to the left or down towards the bottom.] But, the third parameter that allows you to see where you have plotted is invaluable. Use it to help find screen positions.

Here is a program that you may find useful in working out a position. Any time you want to compare your listing on paper and you have a printer, you can send a program listing to the printer with : LIST #8 ...

LISTING for LOCATE:

```
10 REM Locate, Plot and Origin ---
20 MODE 1:CLS
30 s=1:t=0
40 FOR n=1 TO 25
50 LOCATE s,s
60 PRINT "*"s ","";s
70 ORIGIN t,t
80 PLOT t,t,1
90 s=s+1
100 t=t+8
110 NEXT n
120 WINDOW #1,10,20,1,3
130 PRINT #1,"* 1,1 (1)"
140 WINDOW #2,25,35,12,14
150 PRINT #2,"* 1,1 (2)"
160 WINDOW #3,1,10,12,14
170 PRINT #3,"* 1,1 (3)"
180 WINDOW #4,10,20,25,25
190 PRINT #4,"* 1,1 (1)"
200 k$=INKEY$:IF k$="" THEN 200
```

CHAPTER SEVEN COUNTRY ROADS...

The most confusing part I found was understanding colours. With PEN and PAPER shown to be the same, apparently, (see Chapt. 1. Page 50 in Manual) it took me an age to understand what was really happening. Have a look at the title under the chart: PAPER/PEN/MODE/INK reference- Default Settings.

When you turn on the computer, and use the command: PAPER 3 the screen will become bright red, as the computer is in MODE 1. Find the PAPER number 3, move across to the column for MODE 1, and you will see that the INK colour is 6. Refer to the Master Colour Chart on the disc drive. 6 is Bright Red.

To show how the PAPER command works according to the MODE, LOAD"CHARSET and we'll adjust it a little.

LIST the program and Add or EDIT the following lines:

```
10 REM using the PAPER command                        [EDIT and ADD]
20 m=2                          [EDIT and DEL until this is left]
61 FOR p=0 TO 15                                  [ADD a new line]
62 PAPER p                                        [  "   "   "   "  ]
121 PRINT                                         [  "   "   "   "  ]
122 PRINT "Paper";p;"Mode";m                      [  "   "   "   "  ]
123 NEXT p                                        [  "   "   "   "  ]
130              [type the line number to DELETE it completely]
140              [  "   "   "   "      "   "      "    "       ]
200 FOR t=1 TO 2000:NEXT t                  [New time delay line.]
210 MODE 1:END                           [Old line-  new line no.]
```

EXPLANATIONS:

10 Add more information to the REMark line.
20 Stop the INK commands from working.

61 The beginning of a loop to show the 16 PAPER colours. By putting this loop around the outside of another loop, we have created a 'nested loop'- a loop inside a loop ! 'p' is the variable for PAPER.

62 The PAPER colour is to be whatever the value is of 'p'.

121 Leave a blank line ie. PRINT nothing.

122 This statement will show the number of the current PAPER and the current MODE. NB. 3 semi-colons.

123 The end of the loop.

130 No longer needed; done in a different way in Line 122.

140 No longer needed. Lasting two minutes, this program will run right through by itself, without interruption.

CHAPTER SEVEN COUNTRY ROADS...

The most confusing part I found was understanding colours. With PEN and PAPER shown to be the same, apparently, (see Chapt. 1. Page 50 in Manual) it took me an age to understand what was really happening. Have a look at the title under the chart: PAPER/PEN/MODE/INK reference- Default Settings.

When you turn on the computer, and use the command: PAPER 3 the screen will become bright red, as the computer is in MODE 1. Find the PAPER number 3, move across to the column for MODE 1, and you will see that the INK colour is 6. Refer to the Master Colour Chart on the disc drive. 6 is Bright Red.

To show how the PAPER command works according to the MODE, LOAD"CHARSET and we'll adjust it a little.

LIST the program and Add or EDIT the following lines:

```
10 REM using the PAPER command                    [EDIT and ADD]
20 m=2                          [EDIT and DEL until this is left]
61 FOR p=0 TO 15                                 [ADD a new line]
62 PAPER p                                       [ " " " " ]
121 PRINT                                        [ " " " " ]
122 PRINT "Paper";p;"Mode";m                     [ " " " " ]
123 NEXT p                                       [ " " " " ]
130              [type the line number to DELETE it completely]
140              [ " " " " " " " " " ]
200 FOR t=1 TO 2000:NEXT t                  [New time delay line.]
210 MODE 1:END                          [Old line- new line no.]
```

EXPLANATIONS:

10 Add more information to the REMark line.
20 Stop the INK commands from working.

61 The beginning of a loop to show the 16 PAPER colours. By putting this loop around the outside of another loop, we have created a 'nested loop'- a loop inside a loop ! 'p' is the variable for PAPER.

62 The PAPER colour is to be whatever the value is of 'p'.

121 Leave a blank line ie. PRINT nothing.

122 This statement will show the number of the current PAPER and the current MODE. NB. 3 semi-colons.

123 The end of the loop.

130 No longer needed; done in a different way in Line 122.

140 No longer needed. Lasting two minutes, this program will run right through by itself, without interruption.

Check the alterations; there were 9 to make, then: SAVE"PAPER and RUN it.

The first two runs in each MODE are a little untidy, but the dividing lines do go straight across the screen eventually.

If alterations are needed, press ESC twice. If you cannot get writing to appear on the screen, type: MODE 1:PEN 2
Type: RENUM to RENUMber the program. SAVE "PAPER".

To show the Default Colours of MODE 0 together, type in this program. LISTING for MODE: use AUTO

```
10 REM window and paper demo
20 BORDER 15
30 MODE 0
40 r=20:b=25
50 FOR wp=0 TO 7
60 WINDOW #wp,wp+1,r,wp+1,b
70 PAPER #wp,wp
80 CLS #wp
90 PRINT #wp,wp
100 r=r-1
110 b=b-1
120 FOR delay=1 TO 250:NEXT delay
130 NEXT wp
```

Explanations:

10 The REMark line.

20 The BORDER could be 2 flashing colours if you like: BORDER 15,14 but it can be very annoying.

30 The MODE

40 Values of the variables 'r' and 'b' to be used in the WINDOW dimensions.
Right edge is 'r' (only 20 character widths in MODE 0) and Bottom edge is 'b'.

50 Values of wp. It will start at 0 and after 8 loops have the value of 7. ('wp' for Window and Paper- same variable for both.)

60 The WINDOW will be number zero first, and its left edge will be zero plus one, and its right edge will be twenty, and the top will be zero plus one and the bottom will be twenty-five.

The edges are the columns, top and bottom are the rows.

70 The PAPER in WINDOW zero will be zero and the colour then is blue.

80 Clear WINDOW zero. If this line is not added, the new colour

will not be seen.

90 PRINT in WINDOW zero, the number of the PAPER- in this loop, zero.

100 Make 'r' one less in value so that it will move to the left.

110 Make 'b' one less in value so that it will move up from the bottom.

120 This is a pause loop; it slows down the speed of operations.

130 Go and do another loop.

LIST, check,SAVE "MODE , and RUN the program.

Numbered rectangles of colour move towards centre of the screen.

This shows only 8 of the possible 16 colours at once. To see the rest LOAD or LIST the program again, then add these lines:

```
AUTO 140
140 r=20:b=25                                        [COPY line 40.]
150 FOR wp=0 TO 7                                    [  "    "   50.]
160 WINDOW #wp,wp+10,r,wp+10,b            [Moves it further over.]
170 PAPER #wp,wp+8                      [Start at the eighth one.]
180 CLS #wp
190 PRINT #wp,wp+8                           [Number 8 on screen.]
200 FOR delay=1 to 250:next delay
210 NEXT wp
220 k$=INKEY$:IF k$="" TNEN 220          [To keep the prompt away.]
```

LIST, check and SAVE"MODE , and then RUN it. The second set of part rectangles appear to the lower right of the screen. All the default colours of MODE 0 are shown at the same time in sequence.

Another method of giving information to the computer to use, is to store it in the form of DATA. Using the same program again, EDIT LINES 40 and 140, so that they read:

```
40 READ r,b
140 READ r,b
```

Insert lines 35,135 and 230:

```
35 RESTORE 230
135 RESTORE 230
230 DATA 20,25
```

As you will be familiar with the program already, you should be able to see how the lines work in with each other.

EXPLANATIONS:

35 This line lets you run the program more than once, while it is in memory. See line 135 explanation.

40 and 140 The variable names are given 'r,b', and you know what they are for already. The READ command tells the computer to look out for a DATA statement. The first piece of DATA it comes to will belong to 'r' and the second piece to 'b'.

135 Tells the computer that the piece of DATA that it READ in line 230 is needed again. If no RESTORE 230 command was given there would be an error message: DATA exhausted.

230 The DATA must be given in the same order as the variables, and unless you want a sentence in DATA, each item must be separated by a comma.

Check that the program works, and SAVE"MODEDATA

Now more about the PLOT command. It is useful in Graphics to show a starting point, and to show whether you have the correct place before giving a FILL command. This next program shows how to draw using the STEP command and the DRAWR command. If an outline is not drawn accurately with all lines joining, when you try to FILL the area with a colour, the colour will leak out and spoil the effect.

```
10 REM FILL and LEAKS
20 INK 0,1
30 INK 1,24
40 INK 2,20
50 INK 3,6
60 CLS
70 FOR x=400 TO 560 STEP 32
80 PLOT x,150
90 DRAWR 0,-122
100 NEXT
110 FOR y=27 TO 155 STEP 32
120 PLOT 400,y
130 DRAWR 160,0
140 NEXT
150 MOVE 550,35:FILL 3
160 MOVE 550,125:FILL 2
170 PRINT "    This shows that a perfect
 outline      will FILL perfectly and an
inaccurate    outline will leak.
             Press a key to continue.."
180 k$=INKEY$:IF k$="" THEN 180
```

EXPLANATIONS:

10 The REMark line that describes what the piece is about.

20 to 50 names the colours in the 4 PENs of MODE 1. The first

part of the INK command tells the computer which number PEN is to be filled, then a comma, then the number of the INK. These are the Default Colours for MODE 1.

60 Clear the screen.

70 Because PLOT uses x and y coordinates, it is easier to use 'x' as the name of the variable that moves across the screen 'y' as the variable that moves up the screen. Another variable could be used, but to save confusion, it should still have x or y attached, eg 'cx', or 'bx' as variables. Programs run more smoothly if the value of the y coordinate is given first, eg y=250:x=100:PLOT x,y .

In this loop 'x' is to start at 400 pixels to the right and go to 560 pixels in jumps of 32 pixels.

80 PLOT uses the position of 'x' . The first is 400 (the start), then 32 is added on to make 432, then 464, 496, 528, 600. Each one of these points is 150 pixels up from the bottom.

90 DRAWR instructs the GRAPHICS PEN to go to the point plotted and draw a line from it to a point in relation to it. The plotted point becomes the starting point. If a line is drawn to the right or up, it is given positive values. If the line goes to the left or down, it is given negative values.

DRAWR 0, means stay on this position.
The first number moves across, the second moves up; 'up' minus 122 means 'down' 122 pixels. Six lines will be drawn downwards.

To see how this works, you may put in a temporary line. Take this line out by typing: 95 and press RETURN when you are satisfied you can see how this part works.

95 FOR delay=1 TO 250:NEXT delay

100 The go-back line of the loop.

110 In this second loop the 'y' coordinate is used to mark the positions going up from 27 to 155 in jumps of 32 for the beginnings of lines to be drawn across.

120 The lines going down started at 400 across and now the lines across are PLOTted to start at that point and go up in steps of 32 pixels ie 27 is the first position, then +32, etc.

130 DRAWR from the PLOTted point 160 pixels across in a straight line (the zero means neither up nor down).

Line 95 could be copied here as 135 and then removed later.

140 Go back and repeat the loop until it is done.

150 MOVE works the same way. The first number refers to the

number of pixels across and the second refers to pixels up. MOVE works from the last position of the GRAPHICS PEN or from an ORIGIN.

FILL 3 means use the INK that is in PEN 3. As this is in MODE 1 the colour will be bright cyan.

160 The same as 150 except that the INK will be bright red.

170 Print information on the screen.

180 The key press line.

Check and SAVE"LEAKS , then RUN the program. It should live up to its name ! An imperfect block is drawn in the bottom right hand corner of the screen. One square FILLs and another leaks..

Something is wrong in one set of lines. LIST the program again. Set the computer to: AUTO 200

```
200 REM plot the block again higher
210 CLS                                              [COPY 60]
220 FOR x=400 TO 560 STEP 32                         [   " 70]
230 PLOT x,200 240 DRAWR 0,-122                      [COPY 90]
250 NEXT                                             [   " 100]
260 FOR y=72 TO 200 STEP 32
270 PLOT 400,y                                      [COPY 120]
280 DRAWR 160,0                                     [   " 130]
290 NEXT                                            [   " 140]
300 MOVE 550,90:FILL 3
310 MOVE 550,190:FILL 2
320 PRINT "Press a key for a perfect FIL
L.."
330 k$=INKEY$:IF k$="" THEN 330       [COPY 180- adjust numbers]
```

See that it has no errors, SAVE"LEAKS and RUN it. WHAT! AGAIN! The third attempt is correct, and yes you will need to do it as this program is used later. LIST the program. Set to : AUTO 400

```
400 REM the perfect FILL
410 CLS                                          [COPY 210]
420 FOR x=400 TO 560 STEP 32                     [  "  220]
430 PLOT x,200                                   [  "  230]
440 DRAWR 0,-128
450 NEXT                                         [COPY 250]
460 FOR y=72 TO 200 STEP 32                      [  "  260]
470 PLOT 400,y                                   [  "  270]
480 DRAWR 160,0                                  [  "  280]
490 NEXT                                         [  "  290]
500 MOVE 550,90:FILL 3                           [  "  300]
510 MOVE 550,190:FILL 2                          [  "  310]
```

Not too difficult with all the COPYing. Check and SAVE"LEAKS and RUN it.

CHAPTER EIGHT CHECK YOUR TYRES !...

Because I'm really kind hearted, I've taken pity on you finally and decided that this is the right time to give you a treat.

Before we got to the next program in the series, type in this one; the explanation is in the REM statements. This is a program to save you keyboard work. You will need to make a list to see the word and the 'f' key number it works from. To use "SMARTKEY" press Control and the keypad number at the same time. The keypad behaves normally until CONTROL is pressed at the same time. RUN SMARTKEY first then start to type in a new program as before. Do not type NEW after you put SMARTKEY into memory, as it would be removed. Later you can change the words to suit yourself.

ALWAYS SAVE THE PROGRAM BEFORE YOU TRY TO RUN IT.

NO MATTER HOW OFTEN YOU SAVE A PROGRAM BY THE SAME NAME, THERE WILL BE ONLY TWO COPIES OF IT ON THE DISC.

Type: NEW [RETURN. To clear the memory. Square brackets are used for hints and comments only- never program.]

```
AUTO
10 REM SMARTKEY
20 REM struggle with the Manual
30 REM define keys and save fingers
40 ' keys must be in the value of 141 to
 159                                    [See KEY in the Manual]
50 ' A KEY command is given using the nu
mber, a comma, and the words etc to be p
rinted, between quotation marks.
60 ' The second part is a KEY DEF comman
d, followed by the key number of the num
eric pad key (ie the key number 15 not i
ts f number f0), a comma, a zero (for no
 you don't want it auto repeated),      [--continued next line,
                                         don't copy this bit--]
70 'followed by a comma, the number of t
he Key on P.22 Chapt. 7 (User Instr.), a
 comma, a zero, (so that it doesn't work
 off the shift key), a comma, and the ex
pansion key number used with the Key com
mand in the first line. Press CONTROL +
an f KEY.

80 KEY 141,"PRINT"
90 KEY DEF 15,0,128,0,141  [Check twice before pressing RETURN.]
100 KEY 142,"WINDOW"
110 KEY DEF,13,0,129,0,142
120 KEY 143,"LOCATE"
130 KEY DEF 14,0,130,0,143
140 KEY 144,"CHR$()"
150 KEY DEF 5,0,131,0,144
```

```
160 KEY 145,"ORIGIN"
170 KEY DEF 20,0,132,0,145
180 KEY 146,"FILL"
190 KEY DEF 12,0,133,0,146
200 KEY 147,"MOVE"
210 KEY DEF 4,0,134,0,147
220 KEY 148,"INPUT"
230 KEY DEF 10,0,135,0,148
240 KEY 149,"Press a key..."
250 KEY DEF 11,0,136,0,149
260 KEY 150,"k$=INKEY$:IF k$="        [NB Add:    "" THEN    plus
270 KEY DEF 3,0,137,0,150                                  a line no.]
```

Check for errors and SAVE"SMARTKEY. Try it out, rather fun, isn't it ! Line 260 needs "" THEN (line no.) added when used. RUN the program and see that it works- nothing appears on screen when you run it, but press CONTROL and an 'f' KEY and you will see it work.

To use it to put in a new program, do the first few lines without AUTO or you will find the lines of SMARTKEY come up. If they do, CLR the line (or DELete it), and put in the new line- SMARTKEY will keep working. Put it in again each time you reset the computer, or switch it on.

Make a small card cut to the shape of reverse squared brackets- like key 19- to sit around the edge of the keypad. I have a piece of paper taken from a window-envelope sitting at the side of the keypad. With the cellophane removed, the long edge of the envelope window fits around the keypad. I have put the key number and what each key does on the small piece of paper. I can see at a glance which 'f' key + CONTROL to press.

The easiest way to write a program is to take the framework of one and adapt it to another. Perhaps you have noticed that this has been my method. The next two programs are longer than they need be, but by doing them the long way, you can see and prove what happens more easily.

These programs are to help you understand the complexities of the colour commands- and to help you learn more about BASIC.

First, LOAD"LEAKS again. LIST it and delete the following lines:

DELETE 10-330

LIST it to see that you have lines 400-510, and remove line 510.

510 [RETURN]

RENUM and LIST the program now.

It will now run from line 10 to line 110. It is now ready to work into another program.

SAVE"BLOCK

The computer is ideal for mathematical demonstrations. The squared off rectangle just asks to be used to show how to calculate Area.

RUN"SMARTKEY , and then, LOAD"BLOCK. LIST it and AUTO 120

LISTING for AREA:

```
120 LOCATE 4,2:PRINT "Calculating the Ar
ea of a Rectangle"
130 LOCATE 4,3:PRINT "-----------------
-----------------"
140 LOCATE 2,5:PRINT "  The area of a su
rface is measured in    square units.  T
he diagram shows an     area divided int
o squares.  The shaded  area shows one s
quare."
150 LOCATE 26 12:PRINT CHR$(242)+CHR$(15
2);" 5m ";CHR$(146)+CHR$(243)
160 LOCATE 24,14:PRINT CHR$(240)
170 LOCATE 24 15:PRINT CHR$(145)
180 LOCATE 23,17:PRINT "4m"
190 LOCATE 24,19:PRINT CHR$(148)
200 LOCATE 24,20:PRINT CHR$(241)
210 LOCATE 34,22:PRINT CHR$(242)+CHR$(24
3)
220 LOCATE 37,19:PRINT CHR$(240)
230 LOCATE 37,20:PRINT CHR$(241);"1m"
240 LOCATE 34,23:PRINT "1m"
250 LOCATE 2,12:PRINT "Length x breadth"
260 PRINT " gives 20 squares"
270 PRINT " with sides of"
280 PRINT " equal length."
290 LOCATE 2,18:PRINT "Press a key"
300 k$=INKEY$:IF k$="" THEN 300
310 MOVE 422,186:FILL 3:MOVE 454,186:FIL
L 2:MOVE 486,186:FILL 3:MOVE 518,186:FIL
L 2:MOVE 550,186:FILL 3
320 MOVE 422,154:FILL 2:MOVE 454,154:FIL
L 3:MOVE 486,154:FILL 2:MOVE 518,154:FIL
L 3:MOVE 550,154:FILL 2
330 MOVE 422,122:FILL 3:MOVE 454,122:FIL
L 2:MOVE 486,122:FILL 3:MOVE 518,122:FIL
L 2:MOVE 550,122:FILL 3
340 MOVE 422,90:FILL 2:MOVE 454,90:FILL
3:MOVE 486,90:FILL 3:MOVE 518,90: FILL 3
:MOVE 550,90:FILL 2
350 LOCATE 3,23:PRINT "AREA = LENGTH x B
READTH"
360 LOCATE 10,25 :PRINT "5m x 4m = 20m"
370 k$=INKEYS:IF k$="" THEN 370
380 CLS:END
```

EXPLANATIONS:

110 The heading. LOCATE goes down to the line required.

120 Underlining with the hyphen.

130 An explanation to go on screen.

140 Use the CONTROL and f3 to get CHR$(). Put the cursor over the last [)] of the brackets and all you have to do is enter the number. Look in the Manual to see the shape that should appear. The Graphics Character Set is easy to find.

150 to 280 position either symbols or writing on the screen, around the sides of the rectangle. The last 4 lines are to the left of it.

290 The key press loop.

300 MOVE... The position to MOVE to can be found by using the PLOT command. Estimate the coordinates and give a PLOT command. PLOT 422,186,1. [Try it out. RUN"BLOCK and insert line 120: PLOT 422,186,1, and RUN it.] 'FILL' in these next two lines alternates between PENs 2 and 3. Each MOVE is 32 pixels further along the line because the squares were drawn with lines 32 pixels apart. After the top row is FILLed the next row is done.

310 - 330 The same applies.

340 and 350 PRINT more information on the lower screen.

360 The key press loop- used this time to suppress the READY prompt.

370 Clears the screen and ends the program.

Change line 10 by typing:

EDIT 10 and then type:

10 MODE 1:BORDER 11

LIST and check as usual, SAVE"AREA, and RUN it. Correct any errors, SAVE it again and RUN it.

Now you should be able to work out how to DRAWR lines on the screen or make a single rectangle and FILL it. Don't hesitate to experiment as it will help you to learn. Mistakes are part of learning computing. Unless, of course, you're perfect and don't make mistakes !..

If you have Amsword or a similar word processing program, there is an option to go into Basic in their Menu. Go into Basic and

LIST. Stop the listing early in the piece and you will see a section that allows you to DEFINE KEYS just as in SMARTKEY. CLR the example line that works on the f0 key and insert your own address or whatever after the first set of quotation marks.

Then change the other keys to words you use often, or even a sentence. Then type: RUN and Amsword will return to the program. Select SAVE AMSWORD from the menu option, and the Key Definitions will be saved too.

Because I have Amsword, I then keep a copy of the program according to the way I have redefined the keys on discs that I keep for different purposes. For example, I have the disc I'm writing this to with a copy of Amsword with the same KEY DEF as SMARTKEY to save me work too!

The next program is one that uses the AREA program to prove that in MODE 1 there are 4 different PENs and that there are 16 different INKs that can be used to fill those PENs. Each PEN is given a separate command. Although the INK colours are repeated, this program may help explain to you why some PENs produce the same colours as others. For example, in MODE 1, PEN 0 and PEN 4 are both INK colour number 1.

As well the program shows how RND works- the random selector. If writing is done in one colour and the PAPER changes to that colour, writing disappears. If writing is done in different PENs and different INKs, they will not disappear unless they match the PAPER too, and that can happen when random selection is used.

MODE 1 is only allowed 4 colours at once on the screen besides the border colour(s). With a choice of 16 PENs, this program will show you how the PEN, INK, PAPER, MODE table in the Manual works.

LOAD"BLOCK and add line 15:

```
15 INK 0,10:INK 1,0
```

and RUN it.

Notice the effect the green background paper has on the other two unnamed PENs, two and three ? EDIT line 15 again and chose another colour for Pen zero (ie INK 0,..) other than green; leave INK 1,0 alone.

When you have fiddled enough, RUN"SMARTKEY, and LOAD"AREA.

DELETE 10-20 [RETURN] DELETE 190-290 [RETURN] RENUM 100,10

When you LIST the program, the numbering will start from 100. These lines are to go in before the ones from AREA. The AREA program is put to a higher number by RENUM so this can be done.

LISTING for PEN.

```
10 REM PEN, INK, & DEFAULT, MODE 1
20 MODE 1:BORDER 11:INK 0,1:INK 1,24:INK
 2,20:INK 3,6
30 CLS
40 LOCATE 2,1:PRINT STRING$(38,166)
50 FOR s=2 TO 24:LOCATE 1,s:PRINT CHR$(2
02):NEXT s
60 FOR s=2 TO 24:LOCATE 40,s:PRINT CHR$(
202:NEXT s
70 LOCATE 2,25:PRINT STRING$(38,166)
80 LOCATE 8,3:PRINT "Pen, Ink, Fill and
Default."
90 LOCATE 8,4:PRINT STRING$(27,CHR$(160)
)
```

AUTO 100 and make alterations to the lines where necessary:

```
100 FOR x=400 TO 560 STEP 32
110 PLOT x,200
120 DRAWR 0,-128
130 NEXT
140 FOR y=72 TO 200 STEP 32
150 PLOT 400,y
160 DRAWR 160,0
170 NEXT
180 WINDOW #1,3,38,6,12:CLS #1:PRINT #1,
" This is the Default Setting of Pen  an
d Ink colours. Pen 0- No 1 Ink,  Pen 1-
No 24 Ink, Pen 2- No 20 Ink, Pen 3- No 6
Ink, Pen 4- No 1 Ink,etc. Mode 1 allows
4 screen colours and 1 (or 2 flashing) b
order colours."
290 LOCATE 18,14: PEN 2: PRINT "Pen 0":L
OCATE 24,14:PEN 3:PRINT CHR$(243)
200 LOCATE 23,16:PEN 2:PRINT "5":LOCATE
22,18:PRINT "10"
210 LOCATE 18,20:PEN 2:PRINT "Pen 15":LO
CATE 24,20:PEN 3:PRINT CHR$(243)
220 LOCATE 25,22:PEN 2:PRINT "Pen 0 1 2
3"
230 LOCATE 38,14:PEN 2:PRINT "4":LOCATE
38,16:PRINT "9":LOCATE 37,18:PRINT "14"
240 PEN 3:LOCATE 4,16:PRINT "Press a key
"
250 LOCATE 4,17:PRINT "  to continue.."
260 k$=INKEY$:IF k$="" THEN 260
270 MOVE 422,186:FILL 0:MOVE 454,186:FIL
L 1:MOVE 486,186:FILL 2:MOVE 518,186:FIL
L 3:MOVE 550:186:FILL 4
280 MOVE 422,154:MOVE 5:MOVE 454,154:FIL
L 6:MOVE 486,154:FILL 7:MOVE 518,154:FIL
L 8:MOVE 550,154:FILL 9
290 MOVE 422,122:FILL 10:MOVE 454,122:FI
LL 11:MOVE 486,122:FILL 12:MOVE 518,122:
```

```
FILL 13:MOVE 550,122:FILL 14
300 MOVE 422,90:FILL 15:MOVE 454,90:FILL
0:MOVE 486,90:FILL 1:MOVE 518,90:FILL 2:
MOVE 550,90:FILL 3
310 k$=INKEY$:IF k$="" THE 310                    [CLR- new line]
320 CLS #1:PRINT #1, "  The Pen (or Fill         [ "    "    " ]
) number is the first   number in the In
k command:   INK 0,1 - the second number
 is the colour.   But as the Mode allows
 only 4 colours then this is the sequenc
e they will follow."
330 k$=INKEY$:IF k$=""THEN 330                    [New lines...]
340 CLS #1:PRINT #1,"Press a Key to Cont
inue for a change from the default setti
ngs to show  other colour combinations.
 These   changes are made by using the I
nk   command."
350 k$=INKEY$:IF k$="" THEN 350
360 p=RND*3:c=RND*26
370 INK p,c
380 CLS #1:INK 0,c:PRINT #1,"     The ba
ckground is Pen 0.            The headin
g is in Pen 1.             The numbers ar
e in Pen 2.           The arrows are in
Pen 3.        If one part vanishes, then
 another Pen must have the same Ink as P
en 0."
390 ON BREAK GOSUB 410
400 GOTO 350
410 END
```

EXPLANATIONS:

10 The REMark line.
20 Set the MODE and BORDER.
30 Clear the screen.

40 Find two columns in on the first row. PRINT a STRING variable, (38 of them), using character number 166. This puts a trim across the top of the screen.

50 (s referring to side) A loop that starts at (row) 2 and runs down to (row) 24 using the variable 's'. Find column one, and row s(2 to 24) and PRINT character No 202. Then do the next value of 's'. Puts a trim down the the left side of the screen.

60 The same again, only this puts a trim down the right side of the screen.

70 The same as line 40, but puts a trim across the bottom of the screen this time. There is a special reason for not continuing the line down to the last character position- the 40 columns across and 25 rows down position- its a real menace if you forget it, and forces the screen to roll up two lines and PRINTs the READY sign . Avoid that last corner in particular when you

try programming yourself- unless you use a waiting for a key press line.

If you make a WINDOW, make it one line deeper than you plan to use, to prevent screen roll. Whenever you try to put an edging on a screen, those four lines are the easiest way to do it. Type just that much of the program in up to line 70. Make 10 REM edgings and 80 k$=INKEY$:IF k$="" THEN 80. SAVE it separately. Call it EDGES- it will take up 1K, but it will be handy later for your own experimentation.

If you want to have an edging that is complete, the the bottom row will need to be row 24 and so long as the screen and the border are the same colour, the fact that one row is missing will not be obvious.

80 The heading for the screen.

90 Underlining of the heading done with a Graphic Character for a change.

100 to 170 Once again the set of squares inside a rectangle.

180 Information about the screen is placed underneath the heading. The left and right sides of this text WINDOW must not overlap the side trim, so it starts in column 3 and ends in column 38. It starts 6 rows down from the top and ends 12 rows down. The screen is cleared. The information that is sent there must be done with sufficient spaces between words so that they end up looking evenly set out. To count the spaces, count the letters in the next line- this is the way you have to do it when you copy a program from elsewhere. [255 characters in this line]

190 to 230 All concerned with sending words and symbols to screen in the appropriate places to label the rectangle to show which square represents which PEN.

240-250 Another way to present an on screen reminder to press a key.

260 Key press line.

270 Only the FILL numbers have changed: they run in sequence from 0 to 15 in order to prove that there are only 4 colours in MODE 1. The last 4 squares start from zero again.

330 Key press line.

340 More information to screen. Clear the WINDOW first, then write a new message- watch spaces again !

360 The variables I would have liked to use here were 'pen' and 'ink- colour'. Because PEN would have acted as a command word, all I would have got was a Syntax Error Message. So think of 'p' representing 'pen' and 'c' as 'colour'. The RND is the

computers Random selector. To make it work, it must be multiplied by the number of items it is to select from. The 3 is the last number in the loop for the 4 colours of MODE 1 (0 to 3- RND counts from zero) and the 26, therefore represents the choice from 27 colours, (0 to 26 = 27) in the PEN. If more choices were assigned to the PEN, the maximum could only be 15 (0 to 15)- the whole point of this program. RUN the program a few times after it is complete and try changing the 3 to 15 to see what happens.

Look at the coloured squares in the rectangle. Each square was FILLed by a different number PEN. If you increase RND*3 to RND*7 the more chances there are of the background colour remaining the same. With RND*3 (ie 0 to 3 = 4) it changes almost every time. Why would a higher number not change as often?

If RND*3 is changed to RND*16, you will get Improper Argument in 370. It is not the line that the argument is in that has the error (argument means 'computer command' in human terms) but the instructions in line 360 that are wrong- ie a wrong value has been assigned to a variable.

370 The INK command using the two variables.

380 The WINDOW is cleared and six rows of information are PRINTed. See if you can make it sit exactly right on screen- watch the spaces !

390 When the ESCape key is pressed go to line 410- if it isn't pressed then this line just sits waiting until it is...

400 A simple loop that sends the computer back to do another run through the colour change, after a key has been pressed.

410 Stop !

Be very particular about getting the PRINT statements neatly on the screen. A tidy display on the screen is more effective.

Check, LIST, SAVE"PEN and then RUN it. Wait until the writing in the WINDOW is an easy colour to read before pressing ESC and making a correction.

CHAPTER NINE BACK PEDALLING...

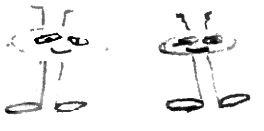
Usually there is more than one way to write a program to get the desired result: it may be very long and complicated in construction, but if it works that is the important thing. If you have written a program and then find that you can see mathematical relationships that you didn't see or couldn't work out before, then try to write the part of the program again that you think you can rewrite more effectively. For example, let's have a look at that MOVE and FILL section in 'PEN'.

LOAD"PEN and DELETE lines 320 onwards: DELETE 320- [DELETE 320 on]

LIST the program so that you are looking at those MOVE and FILL lines, 270 to 300 as you make the changes. These new lines will over write the existing lines; do not use AUTO !!

LISTING for 'PEN-A':

```
270 n=0:x=422:y=186
280 PEN n
290 FOR t=1 TO 20
300 MOVE x,y:FILL n
310 x=x+32
320 IF x=582 THEN y=y-32
330 IF x=582 THEN x=422
340 n=n+1
350 IF n=16 THEN n=0
360 NEXT t
370 k$=INKEY$:IF k$="" THEN 370
380 PEN 3
```



EDIT 10 and put in a new REM line as PEN-A shows another way of writing the MOVE,FILL sequence in 'PEN'.

EXPLANATIONS:

270 n is going to be the PEN number variable. The variables for the MOVE command however can only be x and y. Try other letters of the alphabet- the program will not work because the MOVE command is based on pixel coordinates and recognises only the standard mathematical variables of x for across and and y for up and down. The numbers 422 and 186 were the coordinates mentioned in the first MOVE command.

280 Use the PEN named in the variable 'n'.

290 As there are 20 squares to be FILLed in, go through this section of the program 20 times.

300 MOVE to the x and y coordinates. FILL the area with the PEN colour named in the variable 'n'.

310 Each x coordinate is to shift on by 32 pixels (the STEP was

for 32 when the outline was drawn).

320 The computer does have some funny little ways... Just to be difficult it will not accept an adjustment to the x coordinate first. It lets the last be first instead. So you have to give an instruction to the y coordinate to change before the x. This line tells the point where the coordinates meet, to come back down the screen for 32 pixels (32 was used in the STEP command when the squares were drawn). (This can 'bug' a program.)

330 When x, starting from 422, has enough 32's added on, it will become 582; which would be outside the rectangle. Since we don't want that part of the sceen over-run with leaking colour, when the x coordinate's value increases to 582, then it is time for it to go back to start on another line from 422.

340 Add 1 on to the value of n.

350 This line traps n when its value gets to one more than the number of PENs possible, before it can be acted on. Stop n if it gets to 16 and call it 0.

360 Go through the loop again.

370 The keypress line to suppress the READY prompt.

380 To save you problems, this line tells the computer to use PEN 3, (if you need to make an alteration...by chance). When the program reaches this point, the PEN colour and the background are the same colour, so I've saved you the trouble of searching for what-to-do-when-the-writing-disappears !

Do all the usual things: LIST, SAVE, RUN, correct and SAVE and RUN again, until it works the same as PEN up to that stage. SAVE"PEN-A

Try switching lines 320 and 330 just to see how it doesn't work that way. EDIT line 330 and put an apostrophe REM in the line: 330 'IF x...etc. Create a new line, line 315 and COPY line 330 to it. RUN the program and see how strangely the computer reacts...[DO NOT SAVE THIS version.]

And now not to confuse you, but just to show you that there are several choices when you try to write a program, here is that same section written using DATA. LIST PEN-A and overwrite from line 270 on with these lines:

LISTING PEN-B

```
270 READ n
280 PEN n
290 FOR t=1 TO 20
300 READ x,y
310 MOVE x,y:FILL n
```

```
320 n=n+1
330 IF n=16 THEN n=0
340 NEXT t
350 k$=INKEY$:IF k$="" THEN 350
360 PRINT CHR$(7):PEN 3
370 DATA 0
380 DATA 422,186,454,186,486,186,518,186
,550,186
390 DATA 422,154,454,154,486,154,518,154
,550,154
400 DATA 422,122,454,122,486,122,518,122
,550,122
410 DATA 422,90,454,90,486,90,518,90,550
,90
```

SAVE"PEN-B, and check and RUN it etc.

EXPLANATIONS: [only for lines that are different to PEN-A]

270 READ means go and look for the next lot of DATA, and once you've READ it, cross it out so you don't use it again by mistake. (If the same DATA is to be used twice, the RESTORE DATA command must be used.)

300 READ the values given to x and y, (and cross them off afterwards).

360 Checks if the computer reaches this point [Beeps] and uses Pen 3. Often the only way you can find out where your program is sticking, is to put a 'beep' line in. If the computer PRINTs CHR$(7) ie 'beeps' then the program has worked up to the 'beep'. If no beep sounds, move it back to an earlier part in the program until you isolate exactly where your program goes wrong.

370 The DATA for n. It would have been more trouble to put all the numbers in as they would either have had to be put next to the line or be put at every third position after the x and y DATA. Lines 320 and 330 would have been cut out then, but I think they are easier to put in than 20 numbers and commas.

380 The x and y data in the same order as in the MOVE,FILL of PEN. Use the DATA, cross it out, and go on to the next one. Often all the DATA is together at the end of a program. The READ command sends the computer looking through the program for a DATA line and ignoring all other commands on the way. Then it RETURNs to where it had been.

SAVE, check and then RUN"PEN-B.

If you RUN through the program and try to do it again, you will get a Data Exhausted ... error message.
To overcome this put these lines in:

265 RESTORE 370 [Then you can run through the program

295 RESTORE 380 over and over without error messages.]

From these three examples of the one section of the program you can see that the methods of constructing a program are open to choice, and all are the correct way, so long as it works and you are happy with it. Some ways are easier to key in, and some are easier because they express the ordinary way of thinking rather than the obscure method where lots of variables are used.

The next section is a repeat of combining a number of programs in a Menu. This time you will notice a greater saving of space. Don't skip this section. I think you will have discovered by now that every program has something new in it even if the same basic part is used more than once.

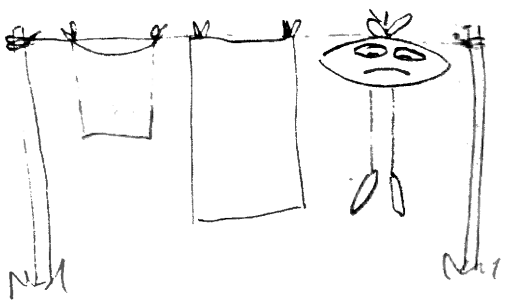
Let me repeat, learning to blend pieces of program into a Menu is the same as writing a large program and blending it together.

LOAD"MENUPLAN and have a look at it again.

Rather bare looking, isn't it? Choose two characters that look interesting side by side and run one plus the other one down each side of the screen. Write the instruction for the two side loops in one line each. Go back and look at the last chapter- the way to do it is described there. Check Answers.
[There are two ways- called MENU1 and MENU1A .]
Where you put the lines makes all the difference. If you don't get the position in the program right the first time, try another position. Remove the old lines if you change them over.

SAVE"MENU1

There are other ways of doing a Menu. These are given in Chapter Fifteen.



CHAPTER TEN TACKLING A ROUNDABOUT..

Because it is important to know how to put things into a menu, we'll take a number of the programs we have done already and put them together in a group. Some of the programs are in different colours, and some in different MODES. To keep the Menu working as it was, the colours and MODE must be part of the actual Menu, not part of a preliminary screen set up. If you put colours and MODE in line 20 and expect it to work the same way after changing to a Graphic Screen and back to the Menu again, you will be in for a messy surprise.

LOAD"MENUPLAN and LIST it.

Do the colour, and screen MODE come in the 5000 plus lines or not? Yes they do. Look at the early lines again.
There is one change needed in line 5010. When MODE 0 is used, the change back to MODE 1 can have a carry over of the last PAPER used in MODE 0. To prevent this, a PAPER command must be placed after the MODE change command and before the CLS command. As well a PEN command will keep the writing the same colour:
EDIT 5010 and add a PEN command:

```
5010 MODE 1:PAPER 0:CLS:PEN 1   [This time SAVE it as SAMPLER.]
```

Next LOAD "CHARSET and LIST it.

RENUM 30,10

The numbering will then be from 30 to 220 in CHARSET.

EDIT 220 and DELete the word END. Insert: GOTO 5000

```
220 MODE 1: GOTO 5000
```

MERGE "SAMPLER and LIST it.

You should then see the two programs have come together without any lines overlapping.

EDIT the following lines and change them to read:

```
5060 PRINT TAB(15), "BASIC SAMPLER"
5070 PRINT TAB(15), "--------------"
5080 PRINT:PRINT TAB(8) "1.   CHARACTER
SET"
5090 PRINT:PRINT TAB(8) "2.   LOCATE(*)
& PLOT(.)"
5100 PRINT:PRINT TAB(8) "3.   PAPER 0-15
IN MODES 0-2"
5110 PRINT:PRINT TAB(8) "4.   WINDOWS, P
APER, MODE 0"
5120 PRINT:PRINT TAB(8) "5.   AS 4, USIN
G DATA"
```

```
5130 PRINT:PRINT TAB(8) "6.   FILL & LEA
KS"
5140 PRINT:PRINT TAB(8) "7.   FILL USING
 16 PENS"
5150 PRINT:PRINT TAB(8) "8. AS 7 -DATA
9. AS 7 -LOOPS"
5160 PRINT:PRINT TAB(12) "0.  END PROGRA
M"

5200 ON INSTR("1234567890",k$) GOSUB 30,
250,550,800,1050,1350,1850,2300,2700,522
0
```

TAB changes in most lines; 0 and new line numbers added to the ON INSTR... GOSUB line.

Test SAMPLER and see that going to No. 1 works and that it returns safely to the menu. Check that you have made all the necessary alterations and then:

SAVE"SAMPLER

LOAD"LOCATE and RENUM 250,10 [RENUM from 250 what was 10.]

LIST the program and add the last line:

530 GOTO 5000 [The next program will start from 550.]

MERGE"SAMPLER

TEST No 2.on the Menu.

SAVE"SAMPLER. Try 1 and 2 and 1 again.

Make any corrections and SAVE the corrected version of SAMPLER.

LOAD"PAPER and make these adjustments:

EDIT Line 240 to read:

240 MODE 1:GOSUB 5000 [To send the program back to the Menu.]

RENUM 550,10

and LIST the program to see what is the last Line number.

(The next program can start from 800 quite safely, when the last line number is 780- even 799.)

MERGE "SAMPLER

TEST the program and then: SAVE "SAMPLER

LOAD"MODE and LIST it..

RENUM 800,10 and add line 1020:

```
1020 GOTO 5000
```

MERGE "SAMPLER Test the program, make any alterations needed and then:

SAVE"SAMPLER

LOAD"MODEDATA and LIST it.

RENUM 1050,10 and add the same line as in the last program:

```
1300 GOTO 5000
```

MERGE "SAMPLER Test the program, make any alterations needed and then:

SAVE "SAMPLER

LOAD"LEAKS and LIST it.

RENUM 1350,10 and add the following lines:

```
1790 FOR delay=1 TO 3000:next delay
1800 GOTO 5000
```

MERGE "SAMPLER Test the program, make any alterations needed and then:

SAVE "SAMPLER

LOAD"PEN and LIST it.

Add these lines (no need to DELete first if they are to be over-written as with 390 to 410).

```
355 FOR t=1 TO 20                          [Allows 20 RND changes.]
385 FOR d=1 TO 200:NEXT d                  [Allows time to see changes.]
390 NEXT t
400 k$=INKEY$:IF k$="" THEN 400
410 GOTO 5000
```

These changes make this program work the same way as the others.

RENUM 1850,10

MERGE "SAMPLER Test the program, make any alterations needed and then:

SAVE "SAMPLER

```
LOAD"PEN-A     and LIST it.

RENUM 2300,19

2680 GOTO 5000

MERGE "SAMPLER      Test the program, make any alterations
                    needed and then:
SAVE "SAMPLER


LOAD"PEN-B      and LIST it.

RENUM 2700,10

2960 n=0
2985 RESTORE 2985
3060
```

DELETE 3060 as the first DATA statement will not work in this Menu.

```
MERGE "SAMPLER          Test the program, make any alterations
                        needed and then:
SAVE "SAMPLER
```

Now for the grand tidy up of the disc: ¦ERA,"*.BAK [bar-ERA] and remove all the files ending in .BAK.
If you want to remove the programs you put in SAMPLER, then:

```
     ¦ERA,"CHARSET
     ¦ERA,"LOCATE
```

and in the same way, remove PAPER, MODE, MODEDATA, LEAKS, PEN, PEN-A, PEN-B.

CHAPTER ELEVEN IN A SPIN...

Many of the things you can do on the computer have formulas that have been worked out ready to use. The way a circle is drawn is regulated by such a formula. You only need to know how to use the formula and not why it is written that way.

A circle needs an ORIGIN and a loop to tell the coordinates COS and SIN how they are to progress. The number given in the coordinates refers to the radius of the circle- the radius is the same in both coordinates, but other information can be added to those same coodinates. Remember the rule of mathematical operation- multiply then add.

This next program will show the ORIGIN of a circle with the radius drawn in. Alongside an identical circle will be drawn using the same ORIGIN as the first, to show how other information can be included in a coordinate. The second circle will be drawn in reverse using PLOT.

The first circle is DRAWn and the second PLOTted. With a small circle using PLOT you get no leaks, but a larger circle may leak when you try to FILL it with colour.

To draw a section of a circle, a complete wedge, you have to consider the direction you are working from in relation to the bottom left corner of the screen, the ORIGIN. If the ORIGIN is MOVEd from the bottom left corner of the screen, the coordinates must still be consistent: all numbers without a preceding sign are considered to have an invisible plus sign and are treated as 'plus' numbers. If they are used for the x coordinate they indicate a move to the right; for the y coordinate a plus number means a move upwards. Minus numbers move left for the x cordinate and down for the y coordinate.

When one coordinate is a plus and the other a minus eg x=100 and y=-50, that would mean go 100 pixels to the right and 50 down- and if the ORIGIN was 0,0(in the bottom left corner) then x=100 and y=-50 would be off the screen!- a legal command.

Where the word ORIGIN is placed can alter what happens. So many functions with the computer depend on them being done in a set order; change the position of two line numbers, and the result can BUG you for weeks. That is the most common BUG- a line in the wrong position. (A BUG is a foul up in a program.)

Occasionally, a line in a different place gives a most unexpected result- the third circle can have very attractive variations- especially when combined with other colours.

LISTING FOR "CIRCLES": [Use SMARTKEY]

```
10 REM circles showing the importance of
 origin
```

```
20 MODE 1:BORDER 4
30 INK 0,18:INK 1,0:INK 2,14:INK 3,7
40 DEG:CLS
50 PRINT:PRINT TAB(10) "DRAW AND PLOT CI
RCLES."
60 PRINT TAB(10) "----------------------
"
70 LOCATE 3,24:PRINT "Origin is the cent
re of left circle."
80 LOCATE 4,25:PRINT "Segments and lace
have new origins."
90 ORIGIN 150,250
100 FOR a=0 TO 360
110 DRAW 100*COS(a),100*SIN(a)
120 NEXT
130 MOVE 0,20
140 TAG
150 PRINT "Radius";
160 TAGOFF
170 MOVE 240,0
180 FOR a=360 TO 1 STEP -1
190 PLOT 340+100*COS(a),100*SIN(a)
200 NEXT
210 MOVE -30,0:FILL 3
220 MOVE 300,0:FILL 2
230 ORIGIN 10,100
240 FOR a=0 TO 45
250 DRAW 100*COS(a),100*SIN(a)
260 NEXT
270 DRAW -0*COS(a),-0*SIN(a)
280 MOVE 10,5:FILL 2
290 ORIGIN 630,100
300 FOR a=135 TO 180
310 DRAW 100*COS(a),100*SIN(a)
320 NEXT
330 DRAW 0*COS(a),0*SIN(a)
340 MOVE -10,5:FILL 3
350 INK 1,0
360 FOR a=0 TO 360
370 ORIGIN 320,135
380 DRAW 100*COS(a),100*SIN(a)
390 NEXT
400 k$=INKEY$:IF k$="" THEN 400
```

EXPLANATIONS:

10 The REMark line.

20 The MODE and the border colour.

30 The colours of the 4 INKs.

40 DEG sets the computer to work in a different form so that it can draw circles. When the computer is first turned on the

Default setting is MODE 1, in 4 colours, and set to the RADians form of working. RAD turns DEG back to radians, just as TAGOFF turns off TAG. Then the screen is cleared.

50 The first PRINT leaves the top line blank, and then the next line gives the heading. When a colon is in a line of program, it indicates the end of one command and the start of another. The new command should be treated as if it was on a separate line. Many of the lines could be grouped together with colons separating the commands. However at this stage it is easier to find errors if the lines are kept shorter.

60 Make the underlining hyphens match.

70 and 80 Information to go on the bottom two lines of screen.

90 The ORIGIN of the x and y coordinates. 'x' is to move 150 pixels to the right and y is to move 250 up from the left lower corner of the screen which is ORIGIN 0,0.

100 The circle loop. If the loop goes from 1 instead of zero, the radius is not drawn as a straight line. The circle is still complete whichever is used, but in this case I want a straight radius line for later in the program. [Try from 1 TO 360 later.]

110 Using the DRAW command, this is the basic formula for making a circle. The radius is 100 pixels.

120 The end of the loop- notice no variable name is given. Usually, 110 to 120 are written as one line, with colons to separate the commands.

130 This MOVEs the Graphics cursor up by 20 pixels and leaves it in line with the ORIGIN point.

140 TAG allows you to write at a point made by the Graphics cursor rather than the standard positions on screen.

150 The label PRINTed on screen.

160 TAGOFF as the name suggests, this switches off TAG.

170 This MOVEs the Graphics cursor to the x and y coordinates in relation to the last ORIGIN given. From the ORIGIN of 150,250 the Graphics cursor is to move 240 pixels to the right and stay at the same level- 0 means neither up nor down.

180 This loop is in reverse and has the step -1. Also, because it is not to show a radius it goes from 360 to 1 rather than 0.

190 The point 240,0 is the centre of the circle, the radius is 100 pixels so the furthest away point of the circle will be 240+ 100 or 340 pixels; therefore the x coordinate of the circle formula has to have that point added on to its position after the formula calculation with the radius has been done. As there

is nothing (0) to add to the y coordinate, nothing is added.

200 The end of the loop.

210 and 220 The first moves the x coordinate left(minus) and the second sends it to the right, to give a position for the FILL command. In both, the y coordinate stays unchanged at 0.

230 A new ORIGIN, working from the bottom left hand corner again, 10 pixels across and 100 up.

240 to 260 The radius is drawn from the centre and one eighth of a circle is drawn. 360/8=45. This is to draw a wedge.

270 This draws a line back to the centre point of the circle (0,0) and completes the wedge.

280 Colours the wedge.

290 A new origin on the far side of the screen.

300 to 340 The same in reverse, but working from a different section of the circle, the 4th eighth.

350 to 390 This is where the position of a line makes all the difference! Put the ORIGIN inside the loop for drawing a circle and you have an attractive creation.

400 The key press line is there to hold the screen.

SAVE"CIRCLES then RUN it. Make any corrections needed and then SAVE it again.

Using lines 350 to 400, change to MODE 0, add INK 0,16 and change the radius of the circle from 100 to 200 (both). Change the ORIGIN to: ORIGIN 320,200 (the centre of the screen).
SAVE"LACE (See answers).

One of the most difficult formulas to understand, is the maths formula for the area of a circle. What does a squared radius look like, when a radius is just the distance from the centre to the circumference of a circle? The next program shows you how the area of a circle is measured.

LISTING FOR "AREA-C"
RUN"SMARTKEY first, then LOAD"CIRCLES and:

Delete lines: 70 [RETURN], 80 [RETURN], DELETE 120- [RETURN]

Set the computer to AUTO and edit the lines as follows:

```
10 REM demonstrate the area of a circle
20 MODE 1:BORDER 19
30 INK 0,16:INK 1,0:INK 2,14:INK 3,7
```

```
40 DEG:CLS                                                    [same]
50 PRINT:PRINT TAB(15) "AREA OF A CIRCLE
."                                                [new TAB and heading]
60 PRINT TAB(15) "-----------------"                 [make hyphens fit]
70                                                        [leave blank]
80 d=45:n=45:s=0                                           [see below]
90  ORIGIN  110,200                [nearer left, and in the centre]
100 FOR a=s TO n                    [variables instead of numbers]
110 DRAW 100*COS(a),100*SIN(a):NEXT          [add colon and NEXT]
120 DRAW -0*COS(a),-0*SIN(a)      [use SHIFT and ARROW,then COPY]
130 n=n+d:s=s+d                                            [new line]
140 NEXT t

Press ESC twice and add:
95 FOR t=1 TO 7                     [Draw 1/8 of a circle 7 times]
```

EXPLANATIONS:

80 }
100 }
130 } The circle is drawn in eight parts, 45 is 1/8 of 360 s is the starting variable, d is the increase, and n is the number controlled by the 1 to 7 loop ie n will become 7 times 45.

LIST, RENUM and RUN "AREAC". You should get 7/8 of a circle.

RENUM on its own means that the first number in the program does not need to be changed, that numbers elsewhere need to be put in to the sequence and numbered in tens.

To complete the circle, two sixteenths need to be drawn in. It is easier (and shorter) to write it in full rather than create the same nested loop used to DRAW the first 7 sections.

The first sections were each 45 parts of the 360 circle; so these will need to be half or 22.5. The starting position will be 7 x 45 or 315 parts around the circle.

If you've done the RENUM, the next line will be number 150:

```
150 FOR a=315 TO 337.5                          [1/16]
160 DRAW 100*COS(a),100*SIN(a):NEXT             [COPY line 110]
170 DRAW -0*COS(a),-0*SIN(a)                    [ "    "   120]
180 FOR a=337.5 TO 360                          [2/16]
190 DRAW 100*COS(a),100*sin(a):NEXT             [COPY line 160]
```

SAVE"AREAC , check it and RUN it.

The next part of the program adds colour to each section. The ORIGIN is still the centre of the circle, so this time you will see how the x and y coordinates in the MOVE command work using negative values to instruct the computer to work to the left for the x coordinate and down for the y coordinate.

Use SMARTKEY to help.

```
200 MOVE 15,10:FILL 2:MOVE 5,10:FILL 3:M
```

```
OVE -5,10:FILL 1
210 MOVE -15,10:FILL 2:MOVE -15,-10:FILL
 3:MOVE -5,-10:FILL 1
220 MOVE 5,-10:FILL 2:MOVE 15,-10:FILL 3
:MOVE 35,-10:FILL 1
```

If you get a really baffling SYNTAX ERROR, it may just be a missing space between words. If a word ends exactly at the edge of the screen and you miss a space between it and the next word, you will find it very hard to pick up, because it will look fine in MODE 1. Change to MODE 2 and LIST the program again and you will find the error easier to pick out. Line 210 is such a line. There is a space before the 3 which can easily be missed.

SAVE, RUN, correct and SAVE again.

In the next part, many of the lines are the same as in the first part, and can be copied (if correct). The circle is going to be taken apart and re-drawn to show it in the form of a rectangle.

```
230 n=314:d=75                          [n for number,d for increase]
240 FOR t=1 TO 4                                      [do 4 times]
250 ORIGIN n,158                              [new ORIGIN position]
260 FOR a=67.5 TO 112.5          [nested loop to draw 1/8 wedges]
270 DRAW 100*COS(a),100*SIN(a):NEXT                [COPY line 190]
280 DRAW -0*COS(a),-0*SIN(a)                       [ "    "   170]
290 n=n+d:NEXT t          [moves the ORIGIN for the next section]
```

Try this section out and if you get 4 wedges drawn like ice cream cones in a row, then you may have it correct.

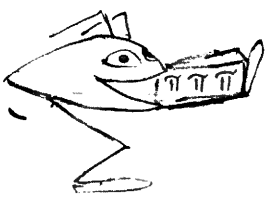
Then SAVE"AREAC and continue. If this piece is not correct, the next part will not fit in, as five wedges are drawn upside down to fit in between those in the first row. Check the numbers you enter! If you stop to try a section out, SAVE as you go!!

```
300 ORIGIN 275,250                                    [new ORIGIN]
310 FOR a=270 TO 293                                   [DRAW 1/16]
320 DRAW 100*COS(a),100*SIN(a):NEXT                [COPY line 270]
                              [Stop to try this out if you wish.]
330 n=350:d=75                          [another value given to n]
340 FOR t=1 TO 3                           [loop to DRAW 3 wedges]
350 ORIGIN n,250                                      [new ORIGIN]
360 FOR a=247.5 TO 292.5              [the loop to make 3 wedges]
370 DRAW 100*COS(a),100*SIN(a):NEXT                [COPY line 320]
380 n=n+d:NEXT t                 [add 75 to n and do another loop]
                              [Stop to try this out if you wish.]
390 ORIGIN 575,250                                    [new ORIGIN]
400 FOR a=270 TO 247.5 STEP -1
410 DRAW 100*COS(a),100*SIN(a):NEXT                [COPY line 370]
                              [Stop to try this out if you wish.]

420 ORIGIN 0,0:MOVE 290,200:FILL 1
                                        [put ORIGIN back to start]
```

```
430 x=300
                                        [start position of x coordinate]
440 FOR t=1 TO 2
                                                  [do this loop twice]
450 FOR n=3 TO 1 STEP -1
                                       [n is used for the Pen colours]
460 MOVE x,200:FILL n
                                  [variables used for position and Pen]
470 x=x+40:NEXT n
                             [add 40 to the position, do the next Pen]
480 NEXT t
                          [end of counting loop, used to make a repeat]
                                    [Stop to try this out if you wish]

490 MOVE 550,200:FILL 2:MOVE 570,200:FIL
L 3                                                          [last 2]
500 LOCATE 20,8:PRINT CHR$(242);"    ";CH
R$(184);"r    ";CHR$(243)
510 LOCATE 17,11:PRINT CHR$(240)
520 LOCATE 17,12:PRINT "r"
530 LOCATE 17,14:PRINT CHR$(241)
540 LOCATE 2,5:PRINT "Circumference = 2"
;CHR$(184);"r"
550 LOCATE 18,20:PRINT "Area = ";CHR$(18
4);" x r x r"
560 LOCATE 22,22:PRINT CHR$(184;" ="3.14
2
570 LOCATE 10,25:PRINT "Press a key to e
nd..."
580 k$=INKEY$:IF k$="" THEN 580
```



500 to 570 Gives information on the screen. Count how many semi-colons and quotation marks in each line and see you have the same.

580 Holds the screen until a key press.

SAVE"AREAC and the RUN it. Make alterations if needed and then SAVE the program again.

You should now be able to see how the area of a circle is calculated.

Using MENUPLAN put AREA and AREAC into a program.

The PAPER 0 command will be needed in line 5010. As both programs have a 'k$=INKEY$:IF k$="" THEN ' line to hold the screen a simple GOTO 5000 is all that will be needed to send the program back to the menu.

CHAPTER TWELVE MOVING FASTER...

To make things appear to move there are some very simple yet effective ways. Colour can create extraordinary illusions. A dark colour will make things look larger and a light colour will make things appear to shrink. If colours are switched quickly using the flashing ability of the computer, things seem to be moving. Try a light and dark colour now- take the extremes black and white, type:

BORDER 0,26 Then quickly type: BORDER 10,24 when the strain gets too much! And then BORDER 11. As I said just light and dark colours give the appearance of movement. RUN"SMARTKEY, then LOAD"AREAC and we'll give movement to that program.

LOAD"AREAC and LIST it, press ESC at line 230:

```
DELETE 150-190
DELETE 230-
```

Now LIST the program and EDIT the lines in:
LISTING for SPINNING

```
10 REM movement through colour.                    [change REMark]
20 MODE 1:BORDER 8                          [change BORDER colour]
30 INK 0,17:INK 1,5:INK 2,14:INK 3,5          [two are the same]
40 DEG:CLS                                          [no change]
50 LOCATE 8,2:PRINT "MOVEMENT USING A CI
RCLE"                                            [new heading]
60 LOCATE 8,3:PRINT "-------------------
----"                                [3 new lines next...]

61 WINDOW #1,1,40,23,25:CLS #1              [information WINDOW]
62 WINDOW #2,21,40,5,21:CLS #2        [right half of the screen]
63 PRINT #1, "   Eight and Sixteen Secti
on Circles"
70 d=45:n=45:s=0                                    [no change]
80 ORIGIN 160,200                                [nearer to left]
90 FOR t=1 TO 8                                  [all 8 sections]
100 FOR a=s TO n STEP 5                            [DRAWs faster]
110 DRAW 100*COS(a),100*SIN(a):NEXT                 [no change]
120 DRAW -0*COS(a),-0*SIN(a)                        [no change]
130 n=n+d:s=s+d                                     [ "   "   ]
140 NEXT t                                          [ "   "   ]
```

RENUM and adjust the MOVE, FILL lines (with new line Nos.).

LIST 180-200 and COPY each line, making these changes:

180 Change the last FILL to 2.
190 Change all the FILL commands to this order: 3,2,3.
200 Place the cursor over the last colon, use the CLR key and hold it down until the last MOVE and FILL commands are gone.

The lines should then read:

```
180 MOVE 15,10:FILL 2:MOVE 5,10:FILL 3:M
OVE -5,10:FILL 2
190 MOVE -15,10:FILL 3:MOVE -15,-10:FILL
 2:MOVE -5,-10:FILL 3
200 MOVE 5,-10:FILL 2:MOVE 15,-10:FILL 3
```

SAVE"SPINNING Now press RETURN and type RUN [RETURN].

You should see the new heading, a circle divided into 8 parts, a lavender BORDER, and a paler lavender screen, and the circle divisions colour up in two alternate shades of blue.
This next line gives the movement:

```
210 INK 2,14,5:INK 3,5,14
```

Now SAVE"SPINNING again. Press RETURN and type RUN .

Effective isn't it? On the opposite side of the screen we will draw a circle with smaller divisions and not only will we make it move, but by making the INK in the Graphics PEN the same colour as the background, show that things can be DRAWn and FILLed without having to be seen.

The DRAW and FILL commands of this next part are repeated. GOSUB is used to send the computer to the following part in the program and RETURN is used to bring it back from where it left.

```
220 GOSUB 260
230 PAPER #2,0:CLS #2
240 GOSUB 270
250 GOSUB 420
260 CLS #1:PEN #1,3
270 d=22.5:n=22.5:s=0
280 FOR t=1 TO 16
290 ORIGIN 480,200
300 FOR a=s TO n STEP 2.5                [DRAWs a little faster]
310 DRAW 100*COS(a),100*SIN(a):NEXT              [  "    "    140]
320 DRAW -0*COS(a),-0*SIN(a):NEXT                [  "    "    150]
330 n=n+d:s=s+d                                  [  "    "    160]
340 NEXT t                                       [  "    "    170]
350 MOVE 20,5:FILL 3:MOVE 20,15:FILL 2:M
OVE 20,40:FILL 3:MOVE 5,30:FILL 2:MOVE -
5,30:FILL 3:MOVE -20,40:FILL 2:MOVE -20,
15:FILL 3:MOVE -20,5:FILL 2

360 MOVE -20,-5:FILL 3:MOVE -20,-15:FILL
 2:MOVE -20,-40:FILL 3:MOVE -5,-30:FILL
2:MOVE 5,-30:FILL 3:MOVE 20,-40:FILL 2:M
OVE 20,-15:FILL 3:MOVE 20,-5:FILL 2

370 LOCATE #1,2,2:PRINT #1, "Please wait
 for the circle to redraw"
380 LOCATE #1,4,3:PRINT #1, "after half
the screen clears.."
```

```
390 FOR delay=1 TO 2000:NEXT delay
400 INK 1,17
410 RETURN
420 PRINT CHR$(7):INK 1,0
430 CLS #1:PEN #1,1:PRINT #1, "With Grap
hics Pen and Paper the same    colour, o
utlines can be drawn invisibly."
440 k$=INKEY$:IF k$="" THEN 440
```

EXPLANATIONS:
220 First this sends the program to clear the information WINDOW and then goes on to draw the second circle in WINDOW 2 with 16 or half size segments. The GOSUB sends the program to line 260 and it goes on down to line 410 before it finds a RETURN to tell it to come back to where it left off and do the next line.

220 Clear WINDOW 2 and make the PAPER the colour of PEN 0 again.

230 This time the GOSUB is to line 270. The information WINDOW is to remain unchanged for the moment. Line 400 changed the Graphics Pen to the same colour as the PAPER. Now all lines are the same as the background and the Heading will have vanished for the moment.
The picture is drawn again, and the program RETURNs to line 250.

250 This GOSUB sends the computer down to line 420, where the bell sounds and the lines and printing turn black. A final piece of information is put in WINDOW 1 after it has been cleared, and the program waits for a key to be pressed to end.

As the lines from line 270 to 360 are identical in function to 100 to 200, they don't need explanation. However, when you do lines 350 and 360, line 350 can be copied using the COPY key, because the only differences between it and 350 are that all the numbers in the MOVE command have the sign changed in front of them. Those without a minus sign get one and those with a minus sign, lose it.

370,380 Information in the WINDOW at the bottom of the screen.

390 The delay line to give you time to look at the screen before it changes.

400 This INK command changes anything written by the Graphics Pen (headings and drawn lines) to the same colour as the PAPER or screen background colour. This can be used to give the appearance of movement as well.

410 RETURN -vital to make a GOSUB work; a GOSUB must always have a RETURN unless the next line leads to the end of the program.

420 The beep and black INK.

430 Last piece of information on screen in the WINDOW.

440 The line to suppress the prompt.

SAVE"SPINNING and RUN the program. Correct it and SAVE again.

The most common way to show movement is by using the cartoon style of movement. Draw a face in the middle of a page, then on a new page, draw it in a new position a little to the left. Repeat this over and over. Put all the pages together and then flick through the pages and the face will appear to move.

This next program called "BLINK" works on that principal. The picture is drawn and coloured in one position and then that is rubbed out (like a page being turned) and is drawn again in a different position, and so on across the screen.

LISTING for BLINK use AUTO .

```
10 REM movement cartoon style
20 BORDER 6
30 WHILE INKEY$=""
40 FOR a=0 TO 640 STEP 20
50 CLS:MOVE a,180
60 DRAW a+40,180,1
70 DRAW a+40,220,1
80 DRAW a,220,1
90 DRAW a,180,1
100 MOVE a+10,182:FILL 3
110 MOVE a,180
120 DRAW a+20,200,2
130 DRAW a+40,180,2
140 FRAME
150 CLS:MOVE a,180
160 DRAW a+60,180,3
170 DRAW a+60,220,3
180 DRAW a,220,3
190 DRAW a,180,3
200 PLOT a+15,210,2:PLOT a+40,210,2
210 FRAME
220 NEXT
230 WEND
```

EXPLANATIONS:
10 The REMark line.

20 Only the outside colour given. If this is put into a menu, you will need to give all 4 INKs from 0 to 3.

30 The start of a loop. Change this to a FOR...TO loop for in a menu.

40 The variable a is going to be used to indicate the pixels across the screen, from 0 to 640- the number of pixels in steps of 20 pixels.

50 Clear the screen, and then bring the Graphics point to 'a' which in the first time through the loop is at 0 pixels across. Go 180 pixels up.

60 This DRAWs a line 40 pixels long from the point 'a' at 180 pixels up, using Graphics Pen 1. After each DRAW command, put in a temporary line so you can see what happens.
65 FOR delay=1 TO 250:NEXT delay (DELETE later.)

70 This DRAWs the left side of a square, using Graphics Pen 1.

80 This DRAWs the top of the square.

90 Bring the line back down to the starting point.

100 Colours the square .

110 Back at the starting point again.

120 This DRAWs a diagonal line up and 130 DRAWs it down to the other corner, using Pen 2.

140 Make the movement less jerky.

150 to 210 DRAWs a rectangle in Pen 3, starting from the same point. The PLOT command is used to mark in the 'eyes'. By making the rectangle wider than the square, it appears to move more.

220 The end of the counting pixels loop.

230 The end of the WHILE loop. Change this line to a match 30 with a next if you put this program in a menu.

Lastly, if you did as I suggested and put a delay line after each DRAW command, remove those lines now and RUN the program.

SAVE"BLINK and then RUN the program.

Have a look at a timer delay loop and see what it does. It takes longer to PRINT on the screen than to go through the timer loops. A beep is sounded at a timed interval.
Type in: DELAY .

```
10 REM watching time fly
20 MODE 2:CLS
30 EVERY 100 GOSUB 80
40 d$="delay"
50 FOR d=1 TO 305
60 PRINT d$;d;     [After SAVE-ing, remove the last semi-colon
70 NEXT d            to see a different result.]
80 PRINT CHR$(7)
90 RETURN       [Unexpected Return in 90 will appear at the end.]
```

Line 30 sends the program to the beep subroutine EVERY 100-counts or almost 2 seconds. SAVE"DELAY and RUN it.

CHAPTER THIRTEEN HOW DO I GET OFF?..

Many programs are designed so that the person using it can choose, within limits, what happens in the program. This is done by using the INPUT command. INPUT asks a question and the answer you give is stored in a VARIABLE.

If the answer to an INPUT must be a number, it can be stored in either a plain VARIABLE (eg n) or a STRING VARIABLE. If the answer must be a letter or a word, or a combination of letters and numbers, then a STRING VARIABLE is needed (eg n$).

INPUT places a question mark on the screen to show that information must be entered.

INPUT is very useful in short programs that help in mathematical claculations. This is where maths are no longer the bane of my life. Once I have got a formula worked out correctly, I can make up a program using INPUT and get the computer to work out my calculations for me. This next program is an example of how to make up such a program.

With INPUT you do not add any signs- eg $ or % when you are answering an INPUT question.

LISTING for PERCENT: [USE SMARTKEY]

```
10 REM-----calculating percentages with
PRINT USING-----
20 MODE 2:CLS
30 LOCATE 30,3:PRINT "CALCULATING PERCEN
TAGE."
40 PRINT TAB(30) "----------------------
-"
50 LOCATE 15,10:INPUT "What is the origi
nal amount (between 1 and 1000.00);b
60 IF b<1 OR b>1000 THEN 20
70 LOCATE 15,12:INPUT "What is the perce
ntage you want to find- omit % sign";f
80 IF f<1 OR f>99.99 then 20
90 r=b*f/100
100 LOCATE 30,18:PRINT r;"is";f;"% of";b
110 LOCATE 35,20:PRINT USING "$$###.##";
r
120 LOCATE 30,25:PRINT "Another calculat
ion Y/N";
130 k$=UPPER$(INKEY$):IF k$<>"Y" AND k$<
>"N" then 130
140 IF k$="N" THEN GOSUB 160
150 GOTO 20
160 END
```

EXPLANATIONS:

10 A comment about the program.

20 Change into MODE 2 with 80 columns across, and clear the screen.

30 Go 30 columns across and 3 rows down and PRINT the heading.

40 In the next line, underline the heading with hyphens.

50 At the 15th column on the 10th row, write a question that requires a number as an answer. Notice that INPUT places a question mark on the screen. The variable here is a plain variable and therefore will only accept an answer if it is a numeral. Try giving a letter of the alphabet as an answer.

60 This is an error trap line. If the answer to the INPUT is not entered correctly- if you enter a number less than 1 or more than 1000- then the program goes back and starts at line 20.

70 In a matching position to line 50, the percntage required is asked for by an INPUT. Take care not to miss the semi-colon before the variable 'f'. The semi-colon is a part of INPUT, and PRINTs the answer you give right beside the question.

80 Another line to trap silly answers. If the number entered is too high or too low, then you are sent back to the beginning again.

90 The formula. 'r' is the variable for 'result', 'b' is the variable for the number you 'begin with', and 'f' is the variable for what you want to 'find'. The variables here were chosen for their association with similar words.

100 This PRINTs out the answer in full. NB 4 semi-colons.

110 PRINT USING can be used in several ways. The most common is to control the setting out of numbers. If you want to limit the size of an answer to be PRINTed on screen, or do accounts, or do a layout using numbers, PRINT USING keeps numbers in line and cuts out long strings of decimals.

The # sign is used to indicate how many character spaces are to be used by a set of numbers- you put in one # for each number column you want printed. If you want two spaces saved for numbers, you put in 2 # signs between double quotation marks. In this program the largest number will need only 3 spaces, so 3 # signs are used before the decimal point. Two decimal places are need, so only 2 # signs go in for them.

To place a $ sign in front of the numbers, two $ signs must be placed inside the quotation marks before the # signs. The semi-colon and the variable for what is to be PRINTed must be beside the last quotation mark, eg PRINT USING "$$####.##";r .

If the number was in the thousands and you wanted commas every 3

numbers , a comma is placed before the decimal point eg: PRINT USING "$$######,.##";r , and r=$10901.10 , the PRINT USING format would make the number appear as $10,901.10 .

120 to 160. These five lines can be used in any program of this kind that you write. All you need to change is the line numbers that the program is sent to. This is the simplest way to cope with a Yes/No [Y/N] choice. Is this information correct Y/N ? Do you want another game Y/N ? This is how to use it correctly.

120 On the bottom line, a question is asked with the choice of Y or N as answers. Try pressing any key other than Y or N. The choice of Y or N being used as the only possible answers is controlled by the next line. The semi-colon is again part of the structure of this command. Did you notice that no variable is required? That is because the Y and the N have been indicated as the only letters to be used to make the program continue, by naming them between the double quotation marks.

130 k$=UPPER$(INKEY$) is a line that turns lower case into upper case. Should you go on and do programs to store lists, this line is a time saver. You can put information in without having to put CAPS LOCK on first, and the information will PRINT out in upper case.

The rest of the line inquires whether Y or N have been pressed. [If k STRING is not equal (<>) to "Y" etc.]

140 Always put in the negative answer decision first. If N is pressed, the computer will act on this line first and go to the line specified to END the program (or it could be sent back to a menu). If Y is pressed, the computer will read this line, see that it doesn't apply and go on to the next line. Don't try to put the next instruction in this line, put in another line to do that.

150 The computer has found that Y was pressed, looked in line 140 to see if there are any instructions for Y there and then found this line had an instruction to follow. It is not necessary to say IF k$="N" first. The send back line must always go to the line that clears the screen.

160 The line to end the program.

SAVE"PERCENT and RUN it.

Try changing this program to one that deals with another mathematical formula. Area of a rectangle for instance.

The next program uses the INPUT command for you to select how many loops you want to see run through in order to trap an error. One of the problems with using nested loops is that if all the parts of the main loop are in variables, the computer adds on an extra one of the variable each time it meets the variable. This can be even more confusing than my explanation-

because you won't be able to see why your program has gone wrong unless you can see how the loops have been working.

Not only will this program show how the nested loops should work, but if some of the lines are put aside with REMs, you will be able to see how the nested loops should not work.

There is also a Printer option. If you don't have a printer just put the line in anyway, but put a REM at the beginning:

210 'PRINT #8,"Outer loop";A [The apostrophe is removed later.]

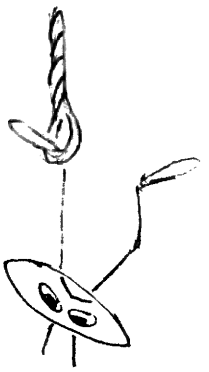
If you don't have a printer, and you don't put in the REM, the program will lock up.

Don't be put off by the length of the program; three quarters of it are lines you can COPY.

LISTING for LOOPRINT: USE SMARTKEY .

```
10 REM There are 9 printer commands to w
atch out for.
20 MODE 2:INK 0,10:INK 1,0:BORDER 15:CLS
30 PRINT TAB(24) "LOOP TRAP -AN AID TO P
ROGRAMMING"
40 PRINT:PRINT "A LOOP can have variable
s for its numbers.  To test the program,
 enter 1 for thefirst number of each loo
p and 2 for the second.  If you do not h
ave a printer  connected, REM out all t
he #8 lines or the program will freeze."
50 PRINT:INPUT "How many loops in the se
ction (2-4)";h
60 IF h<2 OR h>4 THEN 40
70 PRINT:INPUT "The first loop starts at
what number";aa
80 PRINT:INPUT "The first loop ends at w
hat number";a
90 PRINT:INPUT "The second loop starts a
t what number";bb
100 PRINT:INPUT "The second loop ends at
what number";b
110 IF h<3 THEN 180
120 PRINT:INPUT "The third loop starts a
t what number";cc
130 PRINT:INPUT "The third loop ends at
what number";c
140 IF h<4 THEN 290
150 PRINT:INPUT "The fourth loop starts
at what number";dd
160 PRINT:INPUT "The fourth loop ends a
t what number";d
170 IF h=4 THEN 450
180 REM 2 nested loops------
```

```
190 FOR A=aa TO a
200 PRINT "Outer loop";A
210 PRINT #8, "Outer loop";A
220 FOR B=bb TO b
230 PRINT ,"Inner loop";B
240 PRINT #8, ,"Inner loop";B
250 NEXT B
260 b=b-1
270 NEXT A
280 END
290 REM 3 nested loops------
300 FOR A=aa TO a
310 PRINT "Outer loop";A
320 PRINT #8, "Outer loop";A
330 FOR B=bb TO b
340 PRINT ,"Next loop";B
350 PRINT #8, ,"Next loop";B
360 FOR C=cc TO c
370 PRINT , ,"Inner loop";C
380 PRINT #8, , ,"Inner loop";C
390 NEXT C
400 c=c-1
410 NEXT B
420 b=b-1
430 NEXT A
440 END
450 REM 4 nested loops------
460 FOR A=aa TO a
470 PRINT "Outer loop";A
480 PRINT #8, "Outer loop";A
490 FOR B=bb TO b
500 PRINT ,"Next loop";B
510 PRINT #8, ,"Next loop";B
520 FOR C=cc TO c
530 PRINT , ,"Inner loop";C
540 PRINT #8, , ,"Inner loop";C
550 FOR D=dd TO d
560 PRINT , , ,"Most inner loop";D
570 PRINT #8, , , ,"Most inner loop";D
580 NEXT D
590 d=d-1
600 NEXT C
610 c=c-1
620 NEXT B
630 b=b-1
640 NEXT A
650 END
```



EXPLANATIONS:

10 The REMinder line.

20 The MODE, colours and clear the screen.

30 The heading.

40 Information on screen to remind you when you use the program later.

50 The first input, stored in the variable 'h'.

60 If the number is less than two, it won't be a nested loop. If the number is more than 4, the program is not made to take a larger number. This line traps silly answers. Try giving the wrong answer here, to see what happens.

70 The variable 'aa' stores the number the first loop is to start at. The program really should have an error trap line that limits the lowest and highest numbers for each variable, you may wish to put them in. For the sake of brevity they have been omitted. [eg 75 IF aa<0 OR aa>10 THEN 70]

80 The variable a stores the number the loop is to go to. The 'A,aa,a' variables are the Outer loop. The first to start and the last to finish.

90 and 100 the 'B,bb,b' variables form the Next to outer loop. They start second and finish second last.

110 If the number of loops to be done is 2 ie less than 3 then go to line 180. If we said IF h=2 THEN 180, the program would never do more than 2 loops. This is because if h equals 3 then it must equal 2 as well as it is within the limits of line 60, between 2 to 4, which have already been stated.

120 to 160 These can be copied from the lines above. Change the words where necessary.

170 This time it would be possible to say IF h<5 and be correct too. If 'h' is less than 5 it must be equal to 4.

180 This REM line is here so that you can find this section easily in the program if you need to list it and compare how to do this type of loop.

190 'A' is the name of the variable that indicates what name should be used in the matching NEXT loop. 'aa' see line 70. 'a' see line 80

200 This is going to show when the loop happens and 210 is the same line sent to printer. Note that #8 is always followed by a comma in a PRINT statement. PRINT #8, then quotation marks.

220 The 'B' loop is the inner loop in a nested loop.

230 Zones are used here. Note that the line is: PRINT, space, comma, then quotation marks.

240 Zones again. PRINT #8, comma, space, comma, quotation marks.

250 The send back line of the second loop.

260 Take away the value of one 'b'. In the INPUT line the computer gets the value of 'b', then it comes to line 220 and finds another 'b'- that makes two 'b's, so unless we take one away, each time it meets the variable 'b' again, it will add another one on. This is a problem peculiar to variables.

270 The send back line of the first loop.

280 END of the program if only 2 loops were to be done.

The rest of the program duplicates what has been explained already. Take care to get the ZONE commas correct. A plain PRINT has one less comma than a PRINT #8 has.

You may prefer to use LOOPTRAP. It is the same as LOOPRINT with lines deleted and very few other changes. The main difference is that the first number of the loop is given already which may be easier to follow. A printer is not needed.

DELETE lines 70,90,120,150,210,240,320,350,380,480,510,540,570.

EDIT and insert a 1 instead of the double aa,bb,cc,dd in lines 190,220,300,330,360,460,490,520, and 550

LISTING for LOOPTRAP:

```
10 REM no printer needed
20 MODE 2:INK 0,10:INK 1,0:BORDER 15:CLS
30 PRINT TAB(24) "LOOP TRAP -AN AID TO P
ROGRAMMING"
40 PRINT:PRINT "A LOOP with a variable f
or a number must have a reducing line la
ter.   To begin with make the first numb
er 1 and the second 2.  Then you will se
e what happens,otherwise there will be t
oo much for the screen at one time."
50 PRINT:INPUT "How many loops in the se
ction (2-4)";h
60 IF h<2 OR h>4 THEN 40
70 PRINT:INPUT "The first loop ends at w
hat number";a
80 PRINT:INPUT "The second loop ends at
 what number";b
90 IF h<3 THEN 180
100 PRINT:INPUT "The third loop ends at
what number";c
110 IF h<4 THEN 290
120 PRINT:INPUT "The fourth loop ends at
 what number";d
130 IF h=4 THEN 450
140 REM 2 nested loops------
150 FOR A=1 TO a
```

```
160 PRINT "Outer loop";A
170 FOR B=1 TO b
180 PRINT ,"Inner loop";B
190 NEXT B
200 b=b-1
210 NEXT A
220 END
230 REM 3 nested loops------
240 FOR A=1 TO a
250 PRINT "Outer loop";A
260 FOR B=1 TO b
270 PRINT ,"Next loop";B
280 FOR C=1 TO c
290 PRINT , ,"Inner loop";C
300 NEXT C
310 c=c-1
320 NEXT B
330 b=b-1
340 NEXT A
350 END
360 REM 4 nested loops------
370 FOR A=1 TO a
380 PRINT "Outer loop";A
390 FOR B=1 TO b
400 PRINT ,"Next loop";B
410 FOR C=1 TO c
420 PRINT , ,"Inner loop";C
430 FOR D=1 TO d
440 PRINT , , ,"Most inner loop";D
450 NEXT D
460 d=d-1
470 NEXT C
480 c=c-1
490 NEXT B
500 b=b-1
510 NEXT A
520 END
```

SAVE"LOOPTRAP and it is now ready to use. See LOOPRINT for explanations. Once you have seen how the program works by choosing 1 or 2 loops, try the 3 and 4 nested loop combinations. The ESC key will slow the loops down!

CHAPTER FOURTEEN CUTTING CORNERS...

I hope you put in "SCROLL" which was listed in the introduction. Scrolls can be made to run in either direction both in a WINDOW or at a LOCATE-ion across the screen. To get the message moving, a STRING VARIABLE must be used and what is known as STRING SLICING or STRING CUTTING applied.

Although the name sounds daunting, all you really do is count the number of characters in one direction or another and work out how to produce something else. Scrolling uses string slicing by 'slicing' a piece out of a PRINT statement to move it across the screen.

Before we do more with scrolling, this next program is designed to explain the 4 STRING SLICING command words:

LEFT$, RIGHT$, MID$, and LEN($).

LISTING for CUTTING: type AUTO:

```
10 REM LEFT$,RIGHT$,MID$,LEN( $).
20 REM use to cut up string variables.
30 CLS:MODE 2:INK 0,5:INK 1,0 BORDER 11
40 a$="I am too old"
50 b$="my teenage son thinks"
60 c$="to understand"
70 d$=b$+" "+a$+" "+c$
80 PRINT:PRINT TAB(27) "LEFT$,RIGHT$,MID
$,LEN( $)"
90 PRINT TAB(27) "----------------------
---"
100 PRINT:PRINT SPC(8) "a$ says [I am to
o old].";SPC(22) "The LEN(a$) is";LEN(a$)

110 PRINT:PRINT SPC(8) "b$ says [my teen
age son thinks].";SPC(14) "The LEN(b$) i
s";LEN(b$)
120 PRINT:PRINT SPC(8) "c$ says [to unde
rstand].";SPC(22) "The LEN(c$) is";LEN(c
$)
130 PRINT:PRINT SPC(3) "d$  is  b$+a$+c$
plus two added spaces.";SPC(14) "The LEN
(d$) is then";LEN(d$)
140 PRINT:PRINT "If the strings are prin
ted: 'b$',space,'a$',space,'c', then the
 result is:"
150 PRINT:PRINT TAB(17) d$
160 PRINT:PRINT "If we print LEFT$(a$,2)
+RIGHT$(c$,10)+(a space)+LEFT$(b$,2)+MID
$(b$,11,4) then:"
170 PRINT:PRINT TAB(30) LEFT$(a$,2)+RIGH
T$(c$,10)+"  "+LEFT$(b$,2)+MID$(b$,11,4)
180 LOCATE 1,25:PRINT "Press a key to LI
```

```
ST all the program..."
190 k$=INKEY$:IF k$="" THEN 190
200 LIST
```

EXPLANATIONS:
10 and 20 brief notes on the program

30 Clear the screen, go to MODE 2, mauve background, black ink.

40 The contents of 'a' string. Count the number of letters and spaces- ie the characters in the string. This is what LEN does in line 100. There are 12 characters in this string. LEFT$(a$, 2) means that you count the 2 characters from the left, and they will become the LEFT STRING VARIABLE. The 2 characters are [I] -capital I and a space. If it was LEFT$(a$,4), the new string would be [I am]- the first 4 characters from the left.

50 The contents of 'b' string. Count the number of letters and spaces. LEN counts 21 characters in the string in line 110. LEFT$(b$,2) takes the first two characters in b$- [my]. MID$(b$,11,4) works a little differently. Look at b$ and count 11 characters along from the left of the string, take that as the first character and count 4 along. This starts at a space and includes the word son [son].

60 The contents of 'c' string. Count the number of letters and spaces. LEN counts 13 characters in the string in line 120. RIGHT$(c$,10) takes the first 10 characters from the right of the c$- [understand].

70 The contents of 'd' string. LEN counts the three strings and two spaces for the total 48 in line 130.

80 The screen title.

90 The underlining.

100 The first PRINT gives a blank line. SPC( ) is used to set out the screen. The line does what it describes when the LEN(a$) is repeated after the semi-colon.

110 to 130 as in 100.

140 Information on screen.

150 Carries out what was described in 130.

160 Describes what is to be done in line 170.

170 This line combines all the sliced strings to produce another sentence.

180 Information on the bottom line of the screen.

190 The wait for a key to be pressed line.

200 LIST the program on screen.

SAVE"CUTTING then RUN it and correct it, and SAVE it again.

Now back to doing scrolls. They are great for putting a message on the screen. You are forced to look at it because of the movement. Too many scrolls at once, or a scroll moving too quickly can spoil the effect.

Let's have a look at that scroll in the introduction:

LISTING for SCROLL

```
10 REM - a scroll
20 MODE 1
30 INK 0,18
40 INK 1,8
50 a$="I'm starving!  When are we going
to eat? "
60 b$=a$
70 WHILE INKEY$<>" "
80 IF LEN(b$)<40 THEN b$=b$+a$
90 LOCATE 1,1
100 PRINT LEFT$(b$,40)
110 b$=RIGHT$(b$,LEN(b$)-1)
120 WEND
130 END
```

EXPLANATIONS:

10 Note on contents of program

20 The MODE.

30 and 40 The colours.

50 The STRING VARIABLE 'a' is given the words to be printed on screen.

60 Give a$ a second name, or in other words, put the words that are to go on screen into b$ as well. Have a look at line 80. If that was all a$ and no b$, it would be very confusing.

70 The start of a conditional loop. The condition is that while no keys are pressed down, the loop will keep on running.

80 LEN($) measures the characters in a string variable. Here is a chance for you to see just what a 'variable' really is. If you haven't already done so,

LOAD"SCROLL

If you have already RUN"SCROLL, reset the computer by pressing

CONTROL,SHIFT,ESC together and LOAD", but do not RUN" SCROLL.

Type the following:

PRINT LEN(a$) and the screen should show:

0 then type:

PRINT LEN(b$) and the screen should show:

0

Now type RUN. Stop the program after a moment or two by pressing ESC twice and now type those two lines again:

PRINT LEN(a$) and the screen should show:

44 then type:

PRINT LEN(b$) and the screen should show:

a number between 40 to mid 80's!

Try running the program again and stop it at a different place.

PRINT LEN(b$) and PRINT LEN(a$) again.

The length of the a$ will stay the same, but the b$ will be different each time. Because the b$ has the a$ added to it, and depending on how much has crossed the screen, the b$ will be from the same to almost twice as long as the a$.

Back to line 80. If the length of b$ is less than 40 characters then add on a$. 40 characters is the width of a MODE 1 screen.

90 The position on screen where the scroll is to go.

100 The words to go in the scroll must be the same or more than width of the screen allocated to the scroll, in this case 40 characters or more than 40 characters. If the scroll is to be shorter, then you allocate less screen width.

This line says to count the first forty characters from the left in b$ and PRINT them on the screen.

110 Take one away from the length of b$(and you've seen how that is varied by this line) as you count from the right. This is the line that makes the scroll move.

120 The go-and-do-it-again line of the conditional loop. When a key is pressed, the computer will go on to line 130.

130 The program ends and the READY prompt comes on the screen.

If you haven't done so already, SAVE"SCROLL .

Just out of curiosity, put a REM in line 90, the 'LOCATE' line, and RUN the program now. This can be used as a special effect. Optional whether you SAVE that version or not.

Now that you know a little more about writing a program, why not make that program more compact?

Make lines 20,30 and 40 into one line, make lines 50 and 60 into one line, and make lines 90 and 100 into one line. Then type:
RENUM . [See Answers section.]

You must have the lines correctly numbered to continue.

When you have that scroll renumbered and condensed, add these four lines to put a scroll at the bottom of the screen. This scroll will move in the opposite direction. It will run from left to right- which is more difficult to read.

```
35 c$="If I can't have any, neither can
you !  ":c$=d$
75 IF LEN(d$)<40 THEN d$=c$+d$
76 LOCATE 1,25:PRINT RIGHT$(d$,40);              [NB semi-colon]
77 d$=LEFT$(d$,LEN(d$)-1)
```

SAVE"SCROLL-1 and RUN the program.

Refer to the explanation for line 100 on the last page. Notice that this second part is opposite in the use of LEFT$ and RIGHT$ to make the reverse movement. The LENgth of the RIGHT string is counted. This time one character is subtracted from the LEFT string to make the scroll run to the right.

Always try a little experimentation, just to see what happens if..? Take lines 75 and 76 above and alter them a little.

EDIT 75 and then 76.

```
75 IF LEN(d$)<20 THEN d$=c$+d$              [20 instead of 40]
76 LOCATE 3,25:PRINT RIGHT$(d$,38);         [38 instead of 40]
```

This will cause a pause, because the screen is 40 columns wide and line 75 waits until the length is less than 20. Makes the computer think !

Line 76 starts 2 columns in and goes to column 38 across.

SAVE"SCROLLP and experiment with the numbers some more. Try to make the line even smaller, but still balanced.
To show other ways of using scrolls, we will go to MODE 0. This uses an almost identical program. The colours are a little different and so is the wording. Because the screen is only 20 characters wide, the length of the RIGHT$ and LEFT$ are only 20 or less too.

The use of colour in scrolls is effective. If you like you can load SCROLL-1 and use AUTO and EDIT each line or follow the new

listing below. The numbers of the old lines in brackets below:

```
LISTING for SCROLL-0:                                        [old numbers]

10 REM Mode 0 scrolling                                              [10]
20 MODE 0:INK 0,15:INK 1,10:INK 2,6                                  [20]
30 a$="Place your message in line 30...                              [30]
It can be up to 250 characters in length
... ":b$=a$
40 c$="make windows too !    ":d$=c$                                 [35]
50 WHILE INKEY$<>" "                                                 [40]
60 IF LEN(b$)<20 THEN b$=b$+a$                                       [50]
70 LOCATE 1,1:PEN 1:PRINT LEFT$(b$,20)                               [60]
80 b$=RIGHT$(b$,LEN(b$)-1)                                           [70]
90 IF LEN(d$)<20 THEN d$=c$+d$                                       [75]
100 LOCATE 2,25:PEN 2:PRINT RIGHT$(d$,18)                            [76]
110 d$=LEFT$(d$,LEN(d$)-1)                                           [77]
120 WEND                                                             [80]
130 END                                                              [90]
```

EXPLANATIONS:

If you worked from the old numbers, RENUM now.

There are only very slight differences. PENs 1 and 2 are introduced to give 2 colours. Line 30 would need to have the ':a$=b$' removed and made into a new line for there to be 250 characters in the line. The line at present would take only 244 characters in the scroll as 6 characters are taken up by that extra section.

SAVE"SCROLL-0 and then RUN it.

Scrolling in MODE 2 is more difficult to read. Again the last program can be left in the computer, or it can be LOADed and the following lines added. Use the COPY key wherever possible.

LISTING for SCROLL-2

```
10 REM scrolling in a window
20 MODE 2:INK 0,11:INK 1,1
30 a$="Watch this space...You will be ab
le to do scrolling yourself now...Even i
n a window, a location must be used or f
lickering will occur...":b$=a$
40 c$="Scrolling is easier to read in Mo
des 0 and 1...Windows or locations can b
e used...This is a location...":d$=c$
45 WINDOW #1,25,55,12,12:CLS #1                    [new line.]
46 LOCATE 29,3:PRINT "SCROLLING IN MODE            [ "    "  ]
2..."
50 WHILE INKEY$<>" "
60 IF LEN(b$)<30 THEN b$=b$+a$
70 LOCATE #1,1,1:PRINT #1, LEFT$(b$,30)
80 b$=RIGHT$(b$,LEN(b$)-1)
```

```
90 IF LEN(d$)<60 THEN d$=c$+d$
100 LOCATE 10,20:PRINT LEFT$(d$,60)
110 d$=RIGHT$(d$,LEN(d$)-1)
120 WEND
130 END
```

EXPLANATIONS:

Both scrolls run in the same direction this time. If no location is given in a WINDOW the scroll will jump about and seem to flicker- try taking the location half of line 70 away to see what happens. The only new lines are 45 and 46.

45 The WINDOW, a single line deep.
46 A heading at the top of the screen.

Now for a scroll that is sometimes used to end a screen- an up scroll. LOAD"SCROLL-O and EDIT these lines:

LISTING for SCROLLUP

```
10 REM Mode 0 scrolling upwards                          [alter]
20 MODE 0:INK 0,15:INK 1,1:INK 2,6:INK 3                 [  "  ]
,0:INK 4,24
40 c$="Going up !    ":d$=c$
50 WHILE INKEY <>=" "
90 IF LEN(d$)<20 THEN d$=c$+d$
95 FOR n=1 TO 4                                            [new]
100 LOCATE 1,25:PEN n:PRINT RIGHT$(d$,20)                [alter]
110 d$=LEFT$(d$,LEN(d$)-1)
114 FRAME                                                  [new]
115 NEXT n                                                 [new]
120 WEND
130 END
```

SAVE"SCROLLUP and then RUN it. To edit and make corrections if you missed an alteration (eg 3 in line 100) the scroll will not work properly. Lines 30,60,70, and 80 are deleted individually.

Line 114 makes the movement less jerky. Experiment and find what else you can do with a scroll program. The message in this scroll needs to be kept under about 20 characters to be effective. Of course more colours could be used, then the FOR ..TO loop would need to be changed to include any extra INKs.

CHAPTER FIFTEEN PUT IN SOME AIR...

As a storage system the computer can be a most useful instrument. It needs a simple program that asks for the INPUT of information, has a section that will store and retrieve the information, and another section to shows it on screen.

In jargon, such a program is called a Database. You to make a list of information and store it to disc. The information can only be recovered if the same program is used again.

The program is always Menu-driven ie run from a menu. The parts that the menu offer as options all work independently of each other. That is one of the reasons I have tried to show you how to create menus.

The bare bones of a Database are:
1. A menu
2. The ability to make a new list or record.
3. The ability to show or display a record.
4. The ability to load a record from disc.
5. The ability to save a record to disc.

There are several other things you may want it to do, such as make a printed copy, search for a specific record, delete or amend items, or display in a special format on screen. We are going to be concerned with a database that is just a little better than the 'bare bones' type of database.

The Menu that you use can be set out like MENUPLAN or it can be as simple as MENU2.

LISTING for MENU2:

```
10 ' ---another Menu- number 2---
20 'insert dimensions
30 GOSUB 5000
1000 RETURN
2000 RETURN
3000 RETURN
4000 RETURN
5000 '---set up screen---
5010 CLS
5020 PRINT "MENU" :PRINT:PRINT
5030 PRINT "1...ADD"
5040 PRINT "2...DISPLAY"
5050 PRINT "3...LOAD"
5060 PRINT "4...SAVE"
5070 PRINT:PRINT
5080 INPUT "Enter choice..";c
5090 IF c=1 THEN 1000
5100 IF c=2 THEN 2000
5110 IF c=3 THEN 3000
5120 IF c=4 THEN 4000
```

```
5130 IF c<1 OR c>4 THEN CLS:GOTO 5080
```

EXPLANATIONS:

10 The apostrophe is used instead of REM. The dashes are used to make the REMark line of each section stand out .

20 Variables and string variables are used to carry arrays or lists of information. If a variable or a string variable is to be used to hold 10 or more different items of information, the computer must be told that you want a certain number of pigeon holes set aside for the storage of information. Dimensions [DIM] are used to explain the number of 'pigeon holes' required.
DIM NAME$(30),ADDRESS$(30) would let you make a list of 30 names and 30 addresses before the File you created was full.

The array can have more than one dimension, so that the string variable ADDRESS$ could have separate sections in the array for the street,the town and the state. DIM ADDRESS$(30,30,30) would indicate that there were 3 sections or slots to be included in the one 'pigeon hole'.

1000 to 4000 The RETURNs are needed at the end of each section. They are placed here now so that the program will go to the line after a menu choice has been made, and then return to the menu.

5000 on... A very plain menu, with no memory space wasted on frills. If you need to set up a file of sorts, in a hurry, this is adequate.

5080 INPUT is used and the answer stored in a variable- because it is a number. A line for each possible answer must then be given.

5130 The line to prevent mistakes when answering the INPUT. If the number selected is less than 1 or more than 4, go and choose again.

I prefer a slightly better menu myself. Here is a different one that is used in the database program that we are going to work on. RUN"SMARTKEY first.

LISTING for the menu of ADDRESS:

```
10 REM --- address book menu
20 DIM a$(100),b$(100),c$(100),d$(100),e
$(100),f$(100)
30 MODE 2:INK 0,12:INK 1,0
40 LOCATE 14,3:PRINT "ADDRESS BOOK"
50 LOCATE 14,4:PRINT "------------"
60 WINDOW #1,12,36,10,16
70 PRINT #1, "1. Add a record"
80 PRINT #1, "2. Display records"
90 PRINT #1, "3. Search records"
100 PRINT #1, "4. Save file"
```

```
110 PRINT #1, "5. Load file"
120 PRINT #1, "6. Print records"
130 LOCATE 14,22
140 INPUT "Enter choice";c
150 CLS
160 IF c<1 OR c>6 THEN 40
170 ON c GOSUB 220,400,500,620,710,800
180 LOCATE 9,25
190 PRINT "Press any key for Menu.."
200 IF INKEY$="" THEN 200
210 GOTO 30
```

EXPLANATIONS:

The menu is set out so that if you wish to add a trim around it, there is plenty of room to do so.

10 The word 'menu' can be deleted later when this goes into the complete database, but it does describe this section.

20 A program must never be sent back to a DIMension line; the computer will give an error message if the same DIMensions are read twice. The variables used will represent the following: a$ for name, b$ for street (or box no.), c$ for town (or district), d$ for state, e$ for postcode (this will also indicate the state) and f$ for phone numbers. This will allow for 100 names etc. The first array can be stored at 0, but the loop used to handle the arrays goes from 1 to 100.

30 The MODE and INK colours.

40 The heading. How do you find the length of a heading without counting it? Type in line 40 and press ESC twice if you are using AUTO or LIST line 40, and then type:

```
k$="ADDRESS BOOK"   [use the COPY key for heading, press RETURN]
PRINT LEN(k$)

 12
```

Stopping like this in the middle of typing in a program won't upset the program in the least as no line numbers were used. Go back to typing in the program from line 50.

To set out a heading in the centre of the screen you need to know its length. Use the computer to think for you.

50 Underline the heading with hyphens.

60 This makes a WINDOW in the middle of the screen. Again this could have a trim placed around it if you wish.

70 to 120 WINDOW instructions are the hash sign (#), a number and a comma. The titles of the sections of the program are PRINTed in the WINDOW.

130 Position for the INPUT line.

140 The variable 'c' is used for the number to be chosen. An INPUT line always ends with a semi-colon and a variable.

150 Clear the screen ready to go to a new section.

160 If the wrong number is entered, go back and start again, ie less than 1 or more than 6.

170 The line numbers that will be selected by the variable 'c'.

180 The position for a message is marked.

190 The Press a key... message, this will appear at the end of each section that is used in the program.

200 A different way of writing a waiting-for-a-key-to-be-pressed line.

210 This line sends the program back to set up again.

SAVE"MENU3 then RUN the program. You will get a "Line does not exist in 160" if you enter a correct number. A wrong number will clear the screen and the program will start again.

Edit line 160 and send the program back to 20. Run the program and enter 7 as the choice of number. The DIM error message will appear- "Array already dimensioned in 20". If you haven't SAVEd this yet, change line 160 back and then SAVE"MENU3 .

If the file you plan to make is going to be large, include these 3 lines (you are only going to use 3, but put in 4 now):

```
5 OPENOUT "dummy"
6 MEMORY HIMEM-1
7 CLOSEOUT
8 LIST:REM delete this line before you
use this program
```

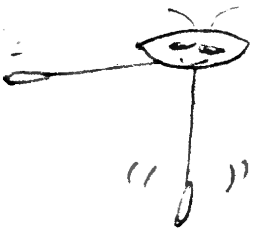
SAVE"MEMORY

If you run MEMORY you are reminded to remove line 8 first. It is there purposely so that something will appear on screen when it is run. If you don't touch the computer for ages and then run SMARTKEY, when you see nothing on screen, aren't you going to wonder what happened to the program? At least you will see MEMORY.

DELETE 8 and MERGE"MENU3

The first stage in using a record keeping program is to start putting in some records.

```
220 REM--- add a record ---
230 FOR n=1 TO 100:CLS
240 IF LEN(a$(n))>0 THEN 370
250 LOCATE 6,10:PRINT "Press [RETURN] to
 end input.":PRINT
260 INPUT " Name";a$(n)
270 IF a$(n)="" THEN 380
280 a$(n)=UPPER$(a$(n))
290 INPUT " Street";b$(n)
300 b$(n)=UPPER$(b$(n))
310 INPUT " Town";c$(n)
320 c$(n)=UPPER$(c$(n))
330 INPUT "State";d$(n)
340 d$(n)=UPPER$(d$(n))
350 INPUT " Postcode";e$(n)
360 INPUT " Phone";f$(n)
370 NEXT
380 LOCATE 10,25:PRINT "No more input."
390 RETURN
```



EXPLANATIONS:

220 This REM line will let you find the section easily when you want to make a slightly different program.. All you need to do is put eg Book Title instead of "Street" and the program can be used for another purpose.

230 The variable 'n' is going to count the records and give each one a number, so that the right address and phone number comes up with the right name!

240 If the length of a$ is nothing, means that if you press return instead of entering a record, the program will go to line 370. This is the signal that you are finished entering records.

250 The information that pressing return will send the program back to the menu, is printed on screen.

260 This first INPUT line can be up to 250 odd characters in length as this array is displayed and printed alone. To use this in another program, change the name of the word to go on screen. The variables have been selected alphabetically, on purpose, so that it will be easy for you to make this program do several different list storage jobs for you.

270 If there is nothing ("" means 'nothing') in the a$(n), means that if you press RETURN instead of entering a record, the program will go to line 380. This is the signal that you are finished entering records.

280 This is a most useful line to use; anything you enter in LOWER CASE or a mixture of UPPER and LOWER CASE is turned into UPPER CASE. Lines 300, 320,340 do the same thing.

290 to 360 These are duplicates of lines 260 and 270. They give the other variables with their INPUT lines. Take great care not to miss any part of ;a$(n) and the other variables at the end of each INPUT line. The first name will be number 1 record in a$(n) ie a$(1) in fact. The first street will be b$(1) and so on.

370 The end of the loop for 'n'.

380 Information PRINTed on screen.

390 This line sends the program back to the menu. The information is stored in the computer's memory, now it needs to know what to do with it. SAVE"ADDRESS

Whenever you are typing in a long program, make a habit to SAVE it at regular intervals. The general rule is to save a program or a file every 15 minutes. The chance of an interruption, power failure or some other disaster can be averted. If an interruption occurs, then only a small amount of program will be lost. I've lost hours of painstaking typing in of a game by not SAVING at regular intervals. One learns...

Test the program now, by running it and selecting 1. Press RETURN for the response. The program should return to the Menu. Press 1 again and try entering a letter to the NAME question. Press RETURN for the rest and RETURN for the next NAME question. If each INPUT has come up, that is as far as you can test the program.

The next stage is to display the information on screen. This program shows all the information in an array for each section, and then rules a line across the screen . Each section of information is PRINTed under the last and the screen rolls upwards as more information is PRINTed out. The ESC key is used to make the list pause, any other will make the list continue.

LISTING to display information on screen.

```
400 REM---display---
410 FOR n=1 TO 100
420 IF a$(n)="" THEN 480
430 PRINT a$(n)
440 PRINT b$(n);"  ";c$(n);"  ";d$(n);"
 ";e$(n)
450 PRINT f$(n)
460 PRINT "------------------------------
-----------"
470 NEXT
480 PRINT SPC(15) "List finished.":PRINT
490 RETURN
```

EXPLANATIONS:

400 Shows where the section of program starts.

410 The start of the FOR..NEXT loop.

420 Again, if the array is empty ie you press RETURN, then this section prepares to return to the Menu.

430 The information stored in a$ is displayed by itself.

440 b$,c$,d$ and e$ make up the address and postcode and are grouped together with 2 spaces (" ") between each section. If this information is too long for one line, it will go on and use another line.

450 The phone number- f$ is also on a line by itself. This is just a suggestion for the arrangement of the display. You may prefer to have each part of the address on a separate line. All you have to do is put each part with a separate line number- there are 9 lines free between 440 and 450 to work on!

460 The dividing line.
470 The end of the loop.

480 Leave 15 spaces, PRINT on screen that the list is finished. After the colon, which shows that this is a new command, PRINT a blank line.

490 Go back to the Menu (actually line 180 to give a key press for the Menu).

Now you will see results. SAVE"ADDRESS then RUN it. Give an answer to each prompt, and after you have entered several sets of names etc., go back to the Menu and select 2. The information you typed in should now be displayed on screen.

The next two sections allow information to be SAVEd to disc and LOADed from disc. As the two sections are almost identical they are listed together.

LISTING for LOAD and SAVE a file.

```
620 REM---save---
630 LOCATE 5,12:INPUT "Name of file to S
AVE:",fs$
640 OPENOUT fs$
650 FOR n=1 TO 100
660 WRITE #9,a$(n),b$(n),c$(n),d$(n),e$(
n),f$(n)
670 NEXT
680 CLOSEOUT
690 LOCATE 20,25:PRINT "File saved."
700 RETURN
710 REM---load---
720 LOCATE 5,12:INPUT "Name of file to L
OAD:",fs$
730 OPENIN fs$
740 FOR n=1 TO 100
750 INPUT #9,a$(n),b$(n),c$(n),d$(n),e$(
n),f$(n)
670 NEXT
```

```
680 CLOSEIN
690 LOCATE 20,25:PRINT "File loaded."
700 RETURN
```

EXPLANATIONS:
Use COPY for most of the 'load' section.
Hash 9 (#9) is the stream used to send information to disc and retrieve information from disc (or tape). When a file is to be written to disc OPENOUT needs a filename, then the file is WRITTEN to disc and CLOSEOUT closes the output. When a file is retrieved from disc, OPENIN needs a filename, then the file is INPUT from the disc and CLOSEIN ends the input.

The actual files do not need to be separated other than by commas see lines 660 and 750. Each section returns to the Menu.
SAVE"ADDRESS.

The last two sections are very much the same as 400 to 490.
LIST 400-490 now so that they are easy to COPY.

```
500 REM---search---
510 INPUT "Find which name";g$
520 g$=UPPER$(g$)
530 FOR n=1 TO 100
540 IF INSTR(a$(n),g$)=0 THEN 590
550 PRINT a$(n)
560 PRINT b$(n);"  ";c$(n);"  ";d$(n);"
 ";e$(n)
570 PRINT f$(n)
580 RETURN
590 NEXT
```

```
800 REM---printer---                    [Watch your numbering!]
810 FOR n=1 TO 100
820 IF a$(n)="" THEN 870
830 PRINT #8,a$(n)
840 PRINT #8,b$(n);"  ";c$(n);"  ";d$(n)
;"  ";e$(n)
850 PRINT #8,f$(n)
860 NEXT
870 RETURN
```

EXPLANATIONS:
510 Asks what name it is to search for, and because we did it earlier, 520 is needed to change the name into UPPER CASE.

540 No name was entered, RETURN was pressed instead.

830 to 840 includes the printer commands, otherwise these sections are the same as lines 400 to 490.

The program is now ready to use. It is easy to alter so that it can be used for more than than just names and addresses. Just change the words 'Name', 'Street' etc and the program is ready to use for another type of file.

CHAPTER SIXTEEN RING YOUR BELL...

Making music on the Amstrad can be tremendous fun. If you read the Manual you may be put off completely! Because the computer has the ability to do all the things described in the scattered notes about making sounds, the Manual seems to overlook the fact that just putting in the melody of a tune is as much as many people will want to do. And fortunately that is quite easy to do.

You will need to have the Manual open at Chapter 7 Page 24 to get the numbers that are used to play the correct notes.

Points to remember-a flat note eg B flat is the same as A sharp. Look up the note you want in the chart under the column NOTE and use the number listed under PERIOD.

Use Channel 1 all the time to do just a melody.
Work from a piece of music so that you get the notes right.
Look at the tempo of the music. Is it in 3/4, or 6/8 time, etc?
Find the shortest note in the piece.
If you want to sing to the music, make the duration of the shortest note be 10 or 15. Use 10 for 6/8, 15 for 3/4, etc.

If the shortest note is a Semi-Quaver (the note has a leg with two strokes at the bottom) and you make the sound last for 15; a Quaver (leg with one stroke) will last for 30; a Crotchet (a note with a leg and a black head) will last for 60; a minim (the note has a leg and an open head) will last for 120; a Semi breve (open head, no leg) will last for 240.

If the music is in 3/4 tempo, then there are 3 crotchets to a bar. Therefore when you do the notes of each bar, their length must equal 3 crotchets or 180. (Duration. 1=0.01 seconds; 180= 1.80 seconds).

If the same note is to be played twice, take 1 away from the first time the note is played and play a 'note of silence', then play the repeated note.

I hope this explanation does not seem too complicated. Use the COPY key for the beginning of each line.

The first listing is the chorus from a march which I think you'll recognise. It has 4 crotchets to the bar, the shortest note is a semi-quaver. A note with a dot after it increases the value of a note by half as much again. A dotted crotchet would get the value of a crotchet plus a quaver.

Beside the first listing I have made extra columns to show more about the notes and how they are used.

Do not copy the NOTE or TYPE and LENGTH columns.

LISTING for "JOHNB" type AUTO 100:
(Type this column only)

```
100 SOUND 1,159,90
110 SOUND 1,179,30
120 SOUND 1,190,45
130 SOUND 1,159,15
140 SOUND 1,119,45
150 SOUND 1,106,15
160 SOUND 1,95,120
170 SOUND 1,119,59
180 SOUND 1,0,1
```

NOTE: TYPE and LENGTH

G	dotted crotchet	90
F	quaver	30
D	dotted quaver	45
G	semi-quaver	15
C	dotted quaver	45
D	semi-quaver	15
E	minim	120
C	crotchet	60
	silence	(1)

Press ESC twice, SAVE"JOHNB and RUN the program- you should recognise the tune.

This first section is repeated. By putting in line 180 now, we save having to write this piece out twice.

Notice line 170. Here is where a minute portion of the length of a note is taken and put into silence in line 180 because the next note is going to be the same as in line 170, in the part which begins at line 310.

```
AUTO 190
190 RETURN
200 SOUND 1,0,60
210 SOUND 1,142,90
220 SOUND 1,127,30
230 SOUND 1,119,45
240 SOUND 1,127,15
250 SOUND 1,119,45
260 SOUND 1,142,15
270 SOUND 1,159,120
280 SOUND 1,190,60
290 SOUND 1,0,60
300 RETURN
```

REST	crotchet	60
A	dotted crotchet	90
B	quaver	30
C	dotted quaver	45
B	semi-quaver	15
C	dotted quaver	45
A	semi-quaver	15
G	minim	120
E	crotchet	60
REST	crotchet	60

Before you RUN this piece, put in the control lines.

```
10 REM chorus of John Brown's Body.
20 GOSUB 100
30 GOSUB 200
40 GOSUB 100
50 GOSUB 310
```

EXPLANATIONS:

10 Tells you what the piece is about.

20 Sends the computer to run through lines 100 to 190 when it meets the command RETURN. Then it comes back to do line 30.

30 Sends the computer to run through lines 200 to 300 when it

meets the command RETURN. Then it comes back to do line 40.

40 Sends the computer to run through lines 100 to 190 when it meets the command RETURN. Then it comes back to do line 50. As we haven't put in line 310 yet, when the computer gets to this point, you will get an Error Message:

Line does not exist in 50.

Now SAVE"JOHNB and RUN the program.

The words are:
Glo-ry! Glo-ry! Hal-le-lu-jah! [3 times]
His soul is march-ing on! [from line 310 to end]

```
AUTO 310
310 SOUND 1,119,60                    C   crotchet          60
320 SOUND 1,106,59                    D   crotchet          60
330 SOUND 1,0,1                           silence           (1)
340 SOUND 1,106,60                    D   crotchet          60
350 SOUND 1,119,60                    C   crotchet          60
360 SOUND 1,127,60                    B   crotchet          60
370 SOUND 1,119,180                   C minim + crotchet   180
380 SOUND 1,0,60                          silence           60
```

SAVE"JOHNB and RUN it.

Just to show that the 1/100 second silence is necessary in line 180, EDIT it now and put an apostrophe REM in it and EDIT line 170 thus:

```
170 SOUND 1,119,60                    [change the 59 to 60]
180 'SOUND 1,0,1                      [put in an apostrophe]
```

Now RUN the tune- do you notice there was a note missing?

Change lines 170 and 180 back the way they were.

The second tune is slightly longer. Again it relies on 1/100 of a second silence between two consecutive notes that are on the same note, so that both notes are sounded. When you see the odd lengths , 14,44,and 59, these are notes that would normally be 15,45, and 60 respectively in length, but have had one deducted from them so that the next note will be sounded.

LISTING for "CLEMENT" Use the COPY key to save typing.

```
10 REM semi-quaver=15,quaver=30,crotchet=60
20 FOR n=1 TO 4                    [the tune is repeated 4 times]
30 SOUND 1,159,44
40 SOUND 1,0,1
50 SOUND 1,159,14
```

```
60 SOUND 1,0,1
70 SOUND 1,159,60
80 SOUND 1,213,60
90 SOUND 1,127,44
100 SOUND 1,0,1
110 SOUND 1,127,14
120 SOUND 1,0,1
130 SOUND 1,127,60
140 SOUND 1,159,59
150 SOUND 1,0,1
160 SOUND 1,159,30
170 SOUND 1,127,30
180 SOUND 1,106,59
190 SOUND 1,0,1
200 SOUND 1,106,60
210 SOUND 1,119,30
220 SOUND 1,127,30
230 SOUND 1,142,119
240 SOUND 1,0,1
250 SOUND 1,142,45
260 SOUND 1,127,15
270 SOUND 1,119,59
280 SOUND 1,0,1
290 SOUND 1,119,60
300 SOUND 1,127,45
310 SOUND 1,142,15
320 SOUND 1,127,60
330 SOUND 1,159,59
340 SOUND 1,0,1
350 SOUND 1,159,45
360 SOUND 1,127,15
370 SOUND 1,142,60
380 SOUND 1,213,60
390 SOUND 1,169,45
400 SOUND 1,142,15
410 SOUND 1,159,119
420 SOUND 1,0,1
430 NEXT n
```

[All lines the same- make a sound on Channel 1, using this note eg 127, and play it for 30/100 of a second.]

[Send back line of loop]

SAVE"CLEMENT and RUN it.

With reasonable care you should be able to make the computer play a simple tune for you now.

For more complete musical sounds you will need to refer to the Manual.

CHAPTER SEVENTEEN MAKING TRACKS...

If you intend to create a program with superb graphics, you will need to know how to design your own characters. We have looked at all the characters available in the CHARSET program. If none of those appeal to you, later you can make up your own.

One or more characters can be placed together to give special effects. Here is the program for a Loading Screen. The numbers of the Graphics Characters have been placed in DATA statements. This program uses existing Graphics Characters.

LISTING for LOADER

```
10 REM a loading screen
20 BORDER 5:INK 0,10:INK 1,0:INK 2,1:INK
 3,6
30 GOSUB 310
40 FOR a=0 TO 2
50 PEN 2
60 LOCATE 12+a,4+b
70 PRINT "A BASIC COURSE"
80 b=b+1
90 NEXT
100 b=0
110 FOR a=0 TO 2
120 LOCATE 14-a,9-b
130 PRINT "for BEGINNERS"
140 b=b-1
150 NEXT
160 b=0
170 FOR a=0 TO 2
180 LOCATE 9+a,14+b
190 PRINT "on the AMSTRAD CPC6128"
200 b=b+1
210 NEXT
220 b=0
230 FOR a=0 TO 2
240 PEN 3
250 LOCATE 14-a,19-b
260 PRINT "by Judith Thamm"
270 b=b-1
280 NEXT
290 PEN 1:LOCATE 15,23:PRINT "Press a key..."
300 GOSUB 520
310 REM draw a screen with data trim
320 CLS
330 RESTORE 550
340 FOR y=1 TO 2
350 FOR x=1 TO 40
360 READ char
370 PEN 1:LOCATE x,y: PRINT CHR$(char);
380 NEXT x,y
390 RESTORE 550
```

```
400 FOR y=24 TO 25
410 FOR x=1 TO 40
420 READ char
430 LOCATE x,y: PRINT CHR$(char);
440 NEXT x,y
450 FOR a=2 TO 22
460 LOCATE 1,a+1:PRINT CHR$(202)
470 LOCATE 3,a+1:PRINT CHR$(202)
480 LOCATE 38,a+1:PRINT CHR$(202)
490 LOCATE 40,a+1:PRINT CHR$(202)
500 NEXT a
510 RETURN
520 k$=INKEY$:IF k$="" THEN 520
530 'CHAIN "daisies"
540 REM data for screen trim
550 DATA 200,198,201,200,198,201,200,198
,201
560 DATA 200,198,201,200,198,201,200,198
,201
570 DATA 200,198,201,200,198,201,200,198
,201
580 DATA 200,198,201,200,198,201,200,198
,201
590 DATA 200,198,201,200,201,196,200,201
,196,200
600 DATA 201,196,200,201,196,200,201,196
,200
610 DATA 201,196,200,201,196,200,201,196
,200
620 DATA 201,196,200,201,196,200,201,196
,200
630 DATA 201,196,200,201,196,200,201
```

EXPLANATIONS:

30 The subroutine PRINTs the characters on screen in a pattern.

60 The location is to move across 3 columns and down 3 rows. In 120 the minus sign moves the locations to the left and down. Try changing 120 to ..9+b.

Watch + and - symbols. Most of this is just copying.

330 So that this can be run more than once without re-loading, this extra RESTORE line has been included.

340 y refers to rows, and x in line 350 to columns. Note y is always given value before x, although the command order is 'x,y'.

360 Because the DATA is character numbers, the instruction to READ DATA must be able to be put into the CHR$() keyword as a variable. The word 'char' is used as a variable to represent the Graphics Characters numbers.

460-490 There have been enough examples of how to use the keyword LOCATE in order to place things in rows, on angles and reverse angles to be able to use one of these yourself. Look at the way LOCATE is used in the examples and use it the same way. Remember that the whole loop needs to be copied so that you don't miss the increase or decrease of a variable eg line 270.

530 This line has the apostrophe REM in it as the program is not available until Chapter 19. This is the way that the CHAIN command is used. After the key press in line 520, the CHAIN command will operate and the named program will run. THE LOADING PROGRAM WILL BE WIPED FROM MEMORY WHEN THE NEXT PROGRAM IS RUN. Leave the apostrophe in until you have SAVED the program! Any Basic program (--------.BAS) could be named between the quotation marks.

550 to 630 Copy line 550 exactly for lines 560,570,580. Lines 590 and 600 are different. Then copy line 600 exactly for lines 610,620 and most of 630.

Making your own Graphics Characters is time consuming. You need a block divided into 8 squares wide and 8 squares deep. Each square has a specific value according to its position. If a square is used, it is given the value of its position, if it is not used, it has no value. Then each line is totalled across and the total is used as part of a program to give a key a new Graphics Character, called a User Defined Character.

```
  128 64 32 16  8  4  2  1
  ------------------------
 !__!__!_*!_*!__!__!__!__! = 48
 !__!_*!__!__!_*!__!__!_*! = 73        This new character
 !__!_*!_*!__!__!_*!__!__! = 100       is a Base Clef sign.
 !__!__!__!__!__!_*!__!_*! = 5
 !__!__!__!__!__!_*!__!__! = 4
 !__!__!__!__!_*!__!__!__! = 8
 !__!__!__!_*!__!__!__!__! = 16
 !__!__!_*!__!__!__!__!__! = 32         [ 8 x 8 GRID ]
```

To put the new character into a program the keywords SYMBOL and SYMBOL AFTER are used.

LISTING for CLEFS:

```
10 REM clefs
20 CLS
30 SYMBOL AFTER 240
40 SYMBOL 241,48,73,100,5,4,8,16,32
50 SYMBOL 242,32,80,72,72,72,80,48,96
60 SYMBOL 243,92,146,170,132,124,4,52,56
70 MODE 1
80 LOCATE 8,12
90 PRINT CHR$(242)
```

```
100 LOCATE 8,13
110 PRINT CHR$(243)
120 LOCATE 8,15
130 PRINT CHR$(241)
```

EXPLANATIONS:

It is usual to start SYMBOL AFTER 240 as 240 is the default setting for User Defined Characters. This allows 16 characters to be defined, from 240 to 255.

Line 40-60 SYMBOL gives the details of the parts to be inked in. The SYMBOL number eg 241 is the number to be used inside the CHR$(241)- character string brackets. The second number is the first line, the third number is the second line, and so on.

The sign for the treble clef is larger and so the characters are designed to be placed one on top of the other. The lines for the music could be drawn in using the method in BLOCK. As well, the Graphics Pen could be used in transparent mode so that both the lines and the character would be seen.

GRAPHICS PEN 1,1 is the command to use for transparent mode.
GRAPHICS PEN 1,0 would turn back to opaque or normal mode.

SAVE"CLEFS and RUN the program.

SYMBOL 242

```
 ----------------------
!__!__!_*!__!__!__!__!__! = 32
!__!_*!__!_*!__!__!__!__! = 80
!__!_*!__!__!_*!__!__!__! = 72
!__!_*!__!__!_*!__!__!__! = 72
!__!_*!__!__!_*!__!__!__! = 72
!__!_*!__!_*!__!__!__!__! = 80
!__!__!_*!_*!__!__!__!__! = 48
!__!_*!_*!__!__!__!__!__! = 96
```

Top of the Treble Clef.

SYMBOL 243

```
 ----------------------
!__!_*!__!_*!_*!_*!__!__! = 92
!_*!__!__!_*!__!__!_*!__! = 146
!_*!__!_*!__!_*!__!_*!__! = 170
!_*!__!__!__!__!_*!__!__! = 132
!__!_*!_*!_*!_*!_*!__!__! = 124
!__!__!__!__!__!_*!__!__! = 4
!__!__!_*!_*!__!_*!__!__! = 52
!__!__!_*!_*!_*!__!__!__! = 56

  128 64 32 16  8  4  2  1
```

Rest of Treble Clef.
[Starts from 32 in row 3.]

Several SYMBOLS can be made to go together to create new shapes.

CHAPTER EIGHTEEN LAST FLING...

By adding a little here and ,putting a variable there, trying a random selection and generally seeing what will come up next, something interesting can be drawn.

The next program relies on repetition, and a slight change of ORIGIN. This could have been written with a menu. Select which cloud you wish to see- No 1, 2 etc. Instead GOSUB has been used to send the computer from one section to the next.

LISTING for CLOUDS: RUN"SMARTKEY first.

```
10 REM clouds
20 INK 0,1:INK 1,26
30 MODE 0:DEG:ORIGIN 0,200:CLG:MOVE 0,20
0
40 GOSUB 100
50 GOSUB 160
60 GOSUB 100
70 GOSUB 170
80 GOSUB 100
90 k$=INKEY$:IF k$="" THEN 90
100 ax=50:y=100:GOSUB 180
110 ax=175:y=150:GOSUB 180
120 ax=250:y=100:GOSUB 180
130 ax=325:y=150:GOSUB 180
140 ax=450:y=100:GOSUB 180
150 ax=525:y=150:GOSUB 180
160 ORIGIN 0,100:MOVE 0,100:RETURN
170 ORIGIN 0,0:MOVE 0,0:RETURN
180 MOVE ax,y+25
190 FOR x=0 TO 360 STEP 10
200 DRAW ax+SIN(x)*50+10*RND,y+COS(x)*25
+10*RND,1
210 NEXT
220 DRAW ax,y:MOVE ax+10,y:FILL 1
230 RETURN
```

EXPLANATIONS:

30 You will need to change back to MODE 1 if you list this program after running it so that you can read it easily. Mode 0 is ideal for designs. DEG sets the computer to cope with circle drawing commands. The ORIGIN is 0, the side line and 200, half way up. The Graphics screen is cleared, and the MOVE 0,200 brings the Graphics Pen up ready to use.

40 to 80 and 100 to 150 are like an automatic menu. With a menu, you press the keys, if you don't feel like pressing keys, put in GOSUBS!

90 This is the last line of the program and stops the READY and CURSOR from appearing.

100 Because two variables are needed for the x values, 'ax' and 'x' are used for the pixels across, y for the pixels up.

160,170 Two different ORIGINs lower on the screen. All the shapes are drawn again.

190 to 220 The section that DRAWs the cloud.

190 The STEP 10 makes this DRAW more quickly.

200 The 50 across and 25 deep makes these wider rather than deep. The final 1 means PEN 1. SIN controls the width of a circle and COS controls the depth. (See more at end of chapter.)

220 In case any of the circles are not complete, another line is DRAWn around to prevent a leak when the FILL is done. The MOVE must be a few pixels away from any of the drawing, or the FILL will not work. Each irregular circular(?) shape starts from the centre. (You can see the line being drawn from the centre.) If the Graphics Pen is still on an already drawn line, the FILL will not work. The RETURN here sends the program back to the main control lines, 40 to 80.

SAVE"CLOUDS and then RUN it.

With just a few alterations, CLOUDS can be made to appear a little different.

LISTING for STONES: Read EXPLANATIONS before you EDIT.

```
10 REM stones
20 INK 0,1:INK 1,26
30 MODE 0:DEG:ORIGIN 0,200:CLG:MOVE 0,20
0
40 GOSUB 100
50 GOSUB 160
60 GOSUB 100
70 GOSUB 170
80 GOSUB 100
90 GOSUB 240                                    [new line 90]
100 ax=50:y=100:GOSUB 180
110 ax=150:y=150:GOSUB 180
120 ax=250:y=100:GOSUB 180
130 ax=350:y=150:GOSUB 180
140 ax=450:y=100:GOSUB 180
150 ax=550:y=150:GOSUB 180
160 ORIGIN 0,100:MOVE 0,100:RETURN
170 ORIGIN 0,0:MOVE 0,0:RETURN
180 MOVE ax,y+25
190 FOR x=0 TO 360 STEP 10
200 DRAW ax+SIN(x)*50+10*RND,y+COS(x)*25
+10*RND,1
210 NEXT
220 DRAW ax,y:MOVE ax+10,y:FILL 1
```

```
230 RETURN
240 LOCATE 1,25:PRINT "Press a key..."
250 k$=INKEY$:IF k$="" THEN 250
260 PEN 0:LOCATE 1,25:PRINT "Press a key
..."
270 FOR n= 1 TO 12:INK 1,n:FOR delay =1
TO 500: NEXT delay:NEXT n
280 k$=INKEY$:IF k$="" THEN 280
290 MODE 2:PEN 1
```

EXPLANATIONS:

Put in the new line 90.

Then EDIT 10,110,130 and 150 and add lines 240 on. These are all the changes you need to make.

Lines 110,130 and 150 place the stones an equal distance apart.

240 on. Press a key... is PRINTed on the bottom of the screen, but not in a WINDOW. To remove the words, they are PRINTed once more, but in the same colour as the background- hence PEN 0. The INKs are changed and a delay loop is used to give time for the changes to be seen.

290 Changes MODE and PEN colour so that the LISTing can be read.

This program could be used as the set up for a stepping stone or guessing game.

SAVE"STONES

Finally, another program based on CLOUDS, but this one is scarcely recognisable. Just change the position of a line and..

LISTING for DAISIES LOAD"CLOUDS and LIST it.

```
10 REM daisies
20 INK 0,1:INK 1,26:2,6
30 MODE 0:DEG:ORIGIN 0,200:CLG:MOVE 0,20
0
40 GOSUB 100
50 GOSUB 160
60 GOSUB 100
70 GOSUB 170
80 GOSUB 100
90 k$=INKEY$:IF k$="" THEN 90
95 GOSUB 290
100 ax=50:y=100:GOSUB 180
110 ax=175:y=150:GOSUB 180
120 ax=250:y=100:GOSUB 180
130 ax=325:y=150:GOSUB 180
140 ax=450:y=100:GOSUB 180
150 ax=525:y=150:GOSUB 180
160 ORIGIN 0,100:MOVE 0,100:RETURN
```

```
170 ORIGIN 0,0:MOVE 0,0:RETURN
180 MOVE ax,y                                   [Remove the '+25'.]
190 FOR x=0 TO 360 STEP 10
200 DRAW ax+SIN(x)*50+10*RND,y+COS(x)*25
+10*RND,1
210 DRAW ax,y:MOVE ax+10,y:FILL 1
220 NEXT
230 GRAPHICS PEN 2,1
240 MOVE ax-10,y+10
250 TAG
260 PRINT CHR$(42);
270 TAGOFF
280 RETURN
290 MODE 2:PEN 1
```

EXPLANATIONS:

EDIT 10. Remove old REM statement and add the new one.

EDIT 20 Add a colon and another INK colour. Later you may wish to change INK 1,26 to INK 1,24 and have yellow daisies. INK 0,11:INK 1,1 is effective too.

Add line 95.

EDIT 180. The program will not work if '+25' is not removed.

EDIT 210. Hold the CLR key until all the word NEXT is swallowed up, then copy the contents of line 220 in its place.

Edit 220. Remove the contents of the line and type in: NEXT . With the position of those two lines changed over the result is quite different! Very effective, all the same.

If this was not the intended result, if I was trying to do the CLOUDS program, you can see how important it is to treat a Bug in a program as first, and most likely, A LINE OUT OF PLACE.

EDIT 230. Remove the word RETURN, then type in the new line. The Graphics Pen is switched to transparent mode- the FILLed background is still visible around the the character, instead of the PAPER background.

240 This MOVE is to position the CHR$(42) to be near the centre when it is PRINTed.

250 TAG allows printing on a graphics screen, and TAGOFF turns the command off.

260 Watch out for semi-colons at the end of lines. They suppress two curly arrows. Leave it off and you will find out.

290 This line is one that can be most useful when you are writing a program. Line 95, the last GOSUB is placed after the hold-the-program-on-the-screen key press line. Then the program

is sent down to line 95. Here the MODE is changed and a visible PEN is called. If you were writing a program, this line could include the program being LISTed:

290 MODE 2:PEN 1:LIST If by chance PEN 1 has the background INK, this line can be given the default INK commands as well. The program will run through to the key press line, and then the MODE changes and the program is LISTed ready for alterations and additions.

SAVE"DAISIES and RUN it, then try the changes I have suggested in the explanations.

Now you can remove the apostrophe REM from line 730 in the LOADER program. The LOADER program can be used as the beginning of a CHAIN of programs. Just another example of BASIC words that are everyday words and easy to use.

When you want to write a program yourself, you will be able to use this book for reference.

As promised, a little more on COS and SIN. LOAD"LACE and alter line 380. Change the radius of COS to 300:
380 DRAW 300*COS(a),200*SIN(a) SAVE"LACE3

Try adding STEP 5 to line 360- SAVE"LACE2 [Try STEP 2.5.]

This should show clearly that COS controls the width of a circle and SIN the depth.

If you don't get the urge to make more changes to 'LACE' besides the alterations I have suggested, I can only say "Experiment!". You should be able to see now how COS and SIN control the shape of a circle.

I can recommend: THE WORKING AMSTRAD by D.Lawrence & S.Lane as an excellent book to progress to for more serious programs. AMSTRAD USERS OMNIBUS by M.Fairbanks has lots of games ideas.

CHAPTER NINETEEN JUST LOOKING...

Amongst the unexplained things that you may have noticed is the term 'USER'. Every time you CAT a disc you will see the word 'USER' followed by a zero. In the Manual, if you have looked the term up, you may have discovered that it is possible to have 16 USER areas.

USER 0 is the normal one to work from, but it is possible to place files in other user areas by using the command: |USER,1 and press RETURN. If you then SAVE a file to that area, it will not appear on the Disc Catalogue. This is because the default CAT area is USER 0. To see the file when you CAT the disc, you will have to change to User 1 [RETURN] and then CAT the disc. [To return to USER 0, either reset the computer or type in |USER ,0]

Also, you may have CAT a disc and found that there are only a few K used, but the disc is reported to have very little free K on it. This is because the other parts of the program may have been placed in different USER areas. Parts of programs can be so thoroughly concealed that even if you do look in all the USER areas, you can not find the actual amount of space the program uses.

NEVER TRY TO PUT ANOTHER PROGRAM ON AN ORIGINAL DISC.

Most original discs have the write protect flap removed, so that you can not accidentally overwrite a program. If you make a back up copy of your original program, then you can experiment with putting more than one program on a disc as the loss of a copy is not a disaster.

This next program allows you to CAT each USER area. Do not RUN the program first, and interrupt it to SAVE the program, because you are likely to SAVE the program into another USER area instead of USER 0. [If you do, you will see it when you RUN the program afterwards. To use the program you must either go into the USER area it is stored in, or change the area it is SAVEd in. |USER,5 (RETURN) eg to go into USER area 5- LOAD the program, type: |USER,0 and then SAVE the program again.]

LISTING for CATUSER:

```
10 REM CAT all USER areas
20 MODE 2:CLS
30 a=0
40 |USER,a
50 CAT
60 a=a+1
70 IF a=16 THEN 90
80 GOTO 40
90 PRINT "CAT another disc Y/N?";k$
100 k$=UPPER$(INKEY$):IF k$="" THEN 100
```

```
110 IF k$="N" THEN 130
120 GOTO 30
130 END
```

EXPLANATIONS:

10 The remark line.

20 If there are very many programs on the disc, MODE 2 gives you a better chance to see them all. Clear the screen.

30 The first value of the variable used to indicate the USER number. This line would not be necessary if the program was only to be used once. The computer gives any variable that does not have its value stated an initial value of 0. Because the program has an option for it to be repeated, the initial value must be stated.

40 The command is always used with the bar sign in front of it, as it is another of the CP/M commands adopted into Basic. The variable 'a' represents the numbers of the areas.

50 CATalogue the disc in the USER area given in line 40.

60 Add one to the value of 'a'.

70 But if the value of 'a' reaches 16, then go to line 90.

80 This is the send back line of the loop that controls the running of the program.

90 The program must be answered with a 'y' or an 'n'.

100 By placing the word UPPER$ in front of INKEY$, and putting INKEY$ in brackets, any answer you give on the keyboard in LOWER CASE will be changed into UPPER CASE.

110 If the answer is 'n'for no, go to line 130 and end the program.

120 If the answer is 'y' for yes, a line is not needed to check that. The program can be made just to run again by sending it back to the line that will make it work in exactly the same way again. In this case line 30 will give the next run through.

There are two CP/M programs which you may wish to use. The first is FILECOPY which is found on side 4 of the Utilities Discs. It is a program that allows you to transfer files or programs from one disc to another when you have only one disc drive. The second is on side 1 of the Utilities Discs and is called PIP. It also copies files or programs from one disc to another, for use with either a one or two disc drives.

FILECOPY:

Insert side 4 of the CP/M discs and type: |CPM . When it has loaded and the A> prompt appeared, type: FILECOPY *.* and press return. The instructions on the screen will tell you what to do then. The '*.*' means any file of any kind; in other words an ambiguous file. Then you are asked for yes or no answers.

If you answer no to the 'Confirm individual files ?', all the disc will be copied. If you answer yes, each file will be named and you will be asked to say yes or no to each one being copied.

PIP:

Insert side 1 of the CP/M discs and type: |CPM . When it has loaded and the A> prompt appeared, type: DIR insert the disc you wanted to copy from, and press return. If you know the names of the files you wanted to copy, you don't need to DIR the disc.

Put the CP/M disc back and type PIP ,the prompt will now become an asterisk- *.

For single and dual disc drives working from the source disc in Drive A, type at the asterisk prompt:

*b:=a: smartkeys.bas	To copy that program from Drive A to Drive B or Drive A to Drive A. You will need to changes discs at the prompts if you have only one drive.
*b:=a: *.*	means copy all programs
*b:=a: *.bas	means copy all programs ending in .bas
*a:=b: menuplan.bas	means copy from Drive B to Drive A- for two disc drives. [NB reverse naming]

The current drive is named at the foot of the screen.

Another handy PIP facility is that if you have a number of files that were done on a word processor and you need copies printed out quickly, PIP will do the job more quickly for you than a Menu driven wordprocessor program. However it will ignore special printer commands such as underlining.

*lst:=myletter.	would send the file saved under eg Amsword to be printed out. If the paper runs out a Retry, Ignore or Cancel message comes across the screen. Insert more paper and press 'I' and the printing out will continue.

A command to print a file can be given more directly. After CP/M has been booted you can type directly:

A>pip lst:=myletter

This is one of the useful business facilities of the CPC6128. In CP/M, CONTROL and P pressed together, will send a text screen to the printer at any time.

DISCKIT3 in detail:

Insert Side 1 of the System/Utilities Discs (the copy) in the disc drive and type: |CPM (the 'CPM' in lower case). This is called 'cold booting'.

```
|CPM
```

```
                          CP/M PLUS
                  Amstrad Consumer Electronics plc

 v.1.0, 61K TPA, 1 disc drive
 A>


                                                  Drive is A:
```

The first rectangle shows the inital screen with title, copyright, version (v. 1.0), the space in the program applications bank (61K TPA), 1 disc drive, the disc drive and prompt A> (drive A and >) the cursor, and the current name of the drive at the bottom of the screen (Drive is A).

At the A> prompt, enter:

```
A> DISCKIT3
```

```
                    Copy                 _7_
                    Format               _4_
                    Verify               _1_
                    Exit from program    _0_
```

The first menu offers 3 choices and a chance to change your mind if you decide not to do anything. The number keys are on the keypad and have an 'f' beside them. f7 will copy and format at the same time. f4 takes you to the Format menu. f1 will verify the format and the state of each track. f0 is the exit key.

Select f7 to format and copy the Master Discs (system format). Select f4 to format most other discs.

```
                    System format        _9_
                    Data format          _6_
                    Vendor format        _3_
                    Exit menu            _._
```

System format, f9 is used for system discs. Format Side A as a System disc, and write 'System' on the A label. Data format f6 is the most commonly used format. Format Side B as a Data disc and write 'Data' on the B label. Vendor format is used for discs that are going to have machine code programs put on them. The full stop on the keypad takes you back to the previous menu.

```
+-------------------------------------------------+
|  ___                                            |
| |   |  Format as Data                           |
| | Y |                                           |
| |___|     Any other key to exit menu           |
+-------------------------------------------------+
```

When you select a format, and before you press Y to confirm that you want a style of formatting, remove the system/utilities disc from the drive and insert the unformatted disc. Then press the Y if it is the format you want. Side A is to be System format, and Side B is to be Data format.

Press any key after formatting both sides of the disc to exit. The first menu will come up again on screen.

```
+-------------------------------------------------+
|                                    _________    |
|          Copy                     |_7_|     |   |
|          Format                   |_4_|     |   |
|          Verify                   |_1_|     |   |
|          Exit from program        |_0_|_____|   |
+-------------------------------------------------+
```

Press the f0 to exit from the program and go back to the CPM prompt A>. You can then go to another CP/M program or reset and return to Basic.

CHAPTER TWENTY ANSWERS...

Chapter One:

a. 2, b. 9, c. 21, d. 45, e. 16, f.9, g. 9, h. 3.

Chapter Nine: The trim down side of a screen page.[2 possible]

MENU1 [title on the disc]

```
5175 FOR a=1 TO 23:LOCATE 1,1+a:PRINT CH
R$(247)+CHR$(246)
5176 FOR a=1 TO 23:LOCATE 39,1+a:PRINT C
HR$(247)+CHR$(246)
```

MENU1A [title on the disc]

```
5175 FOR s=2 TO 24:LOCATE 2,s:PRINT CHR$
(247)+CHR$(246)
5176 FOR S=2 TO 24:LOCATE 38,s:PRINT CHR
$(247)+CHR$(246)
```

Chapter Eleven: Listing for LACE:

```
350 MODE 0:DEG:INK 0,16:INK 1,0
360 FOR a=0 TO 360
370 ORIGIN 320,200
380 DRAW 200*COS(a),200*SIN(a)
390 NEXT
400 k$=INKEY$:IF k$="" THEN 400

SAVE"LACE
```

Chapter Fourteen:

After the lines have been edited:
DELete lines: 30,40,60,100, then RENUM and the result will be:

LISTING for SCROLLA [on the disc as SCROLLA.]

```
10 REM - answer scroll
20 MODE 1:INK 0,18:INK 1,8
30 a$="I'm starving !... When are we goi
ng to eat ?... ":b$=a$
40 WHILE INKEY$<>" "
50 IF LEN(b$)<40 THEN b$=b$+a$
60 LOCATE 1,1:PRINT LEFT$(b$,40)
70 b$=RIGHT$(b$,LEN(b$)-1)
80 WEND
90 END
```

First edition prepared by:
Abbott Copy Centre,
55-57 Gilbert St,
Adelaide. 5000
South Australia.

ISBN 0 7316 0939 5

Written on an Amstrad CPC 6128 computer using Amsword.
First printed on an Amstrad DMP-2000.

Editorial assistance by Percy Cook.
Cover assisted by Reg Pye.